UNIVERSIDAD DE LIMA | 60 años

FACULTAD DE INGENIERÍA Y ARQUITECTURA
CARRERA DE INGENIERÍA DE SISTEMAS

UNIDAD 4:
ESTIMACIÓN DE COSTOS

INGENIERIA DE SOFTWARE I

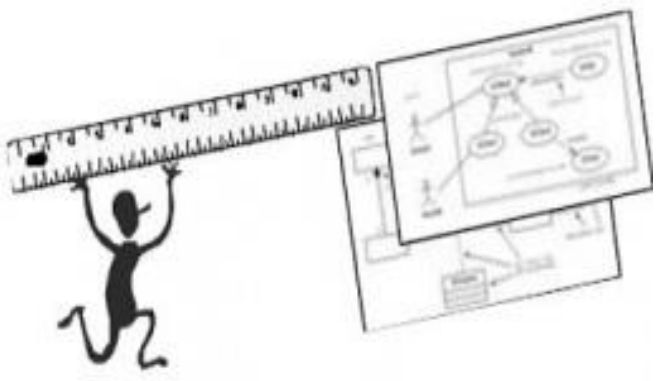
2020 | 2020-1

Agenda

1. Introducción
2. Estimación con Puntos de Función (PF).
3. Puntos de Casos de Uso (PCU)
4. Cocomo 2

Introducción

El proceso de gestión de proyectos de software comienza con un conjunto de actividades que se denominan ***planificación del proyecto***. La primera de estas actividades es la ***estimación*** . Estimar, o *cuantificar*, software no es una tarea fácil.



Antecedentes

- Las metodologías de desarrollo de sistemas han evolucionado, desde los antiguos sistemas por lotes, la programación estructurada, hasta la orientación a objetos, pero las técnicas de estimación no lo han hecho.
- Por un lado, se conocen algunas técnicas como COCOMO, **Puntos de Función** y otras, pero la mayoría de ellas se basan en criterios muy poco efectivos de aplicar como líneas de código, experiencia previa sobre sistemas similares, etc.
- De esta forma, hemos podido notar un cambio en las metodologías para el desarrollo de software y de la misma manera, existen técnicas adecuadas que nos permiten realizar estimaciones, como la **técnica de puntos de casos de uso**, la cual se basa en metodologías orientadas a objetos, específicamente en el **modelo de casos de uso.**

Estimación

- El tiempo para completar un proyecto se ve afectado por lo siguiente:
  1. Número de pasos para completar el caso de uso
  2. Número y complejidad de los actores
  3. Los requerimientos técnicos del caso de uso, como: concurrencia, seguridad o rendimiento
  4. Varios factores ambientales tales como: experiencia y conocimientos del equipo de desarrollo
- Un método de estimación que tome en cuenta tales factores en el ciclo de vida de un proyecto será muy útil para calcular: tiempo, costo y asignación de recursos

Métrica de los Puntos de Función

- Es una métrica que se puede aplicar en las primeras fases de desarrollo.
- Se basa en características fundamentalmente **"Externas"** de la aplicación a desarrollar.
- Mide dos tipos de características:
  - Los elementos de función como entradas, salidas, ficheros, etc.
  - Los factores de Complejidad.

Elementos de Función

- Son elementos fácilmente identificables en los diagramas de especificación del sistema. (DFD, Entidad-Relación, DD)
- Los usuarios los entienden perfectamente.
- Observamos la aplicación como una caja negra.

Punto de Función

El objetivo es traducir en un **Número** el tamaño de la funcionalidad que brinda un producto de software

- Desde el Punto de vista del usuario
- Suma ponderada de características del producto:

Transacciones

- Nro. Entradas Externas (EI- External Input)
- Nro. Salidas Externas (EO- External Output)
- Nro. Consultas Exts. (EQ- External inQuiry)

Datos

- Nro.Archivos Int. Lógicos (ILF- InternalLogical File)
- Nro.Arch. Interfaz Externa (EIF-External Interface File)

Visión de caja negra

- Nos centramos en característica visibles del objeto en estudio.
- Ejemplo:
  - Equipo de música.
  - Coche
  - Animales

Elementos de función

- Entradas
- Salidas
- Consultas
- Ficheros Lógicos o Internos
- Ficheros de Interfaz

Definiciones

- Proceso elemental
- Datos e información de control
- Lógica de proceso
  - Ediciones, algoritmos o cálculos
  - Accesos a ficheros para consulta o actualización

Proceso elemental:

- **Proceso elemental :** Menor unidad de actividad que tiene sentido para el usuario, conocedor del sistema en estudio.

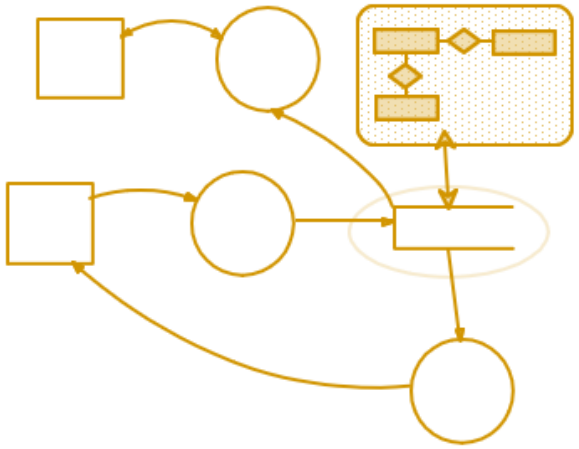
- **Datos e información de control:** Son datos elementales con los que trabaja la aplicación en estudio. Nos referimos a ellos siempre como datos aunque se componen de los datos propios del sistema en estudio, más las informaciones de control que solicita el usuario: mensajes de error, claves de seguridad... etc.

- **Lógica de proceso:** Procesos que se producen como consecuencia de un proceso elemental. Pueden ser de dos tipos:
  - Ediciones, algoritmos o cálculos
  - Accesos a un fichero para consulta o actualización.

Ficheros Lógicos o Internos - ILF

Es un grupo de datos o de información de control, lógicamente relacionado, identificable por el usuario y mantenido dentro de la frontera de la aplicación. Agrupaciones de datos, tal y como los percibe el usuario.

- Es diferente de:
  - Entidades y Relaciones
  - Tablas o archivos resultantes del diseño físico
- Los grupos de datos serán accedidos y actualizados por la aplicación

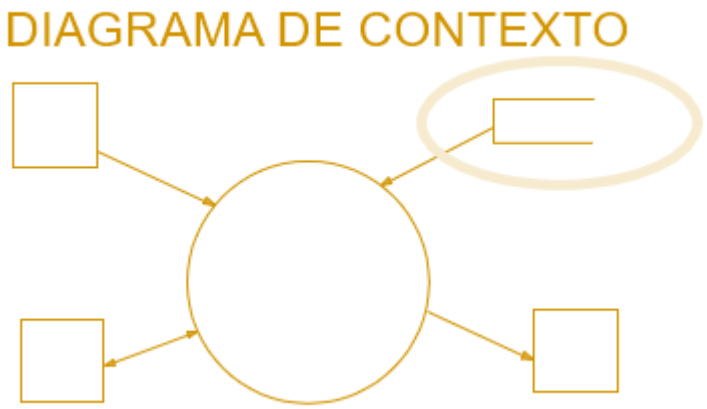


DIFICULTAD FICHEROS LÓGICOS	Número de Campos o Atributos		
	1-19 Atributos	20-50Atributos	51 + Atributos
1 Registro Lógico	BAJA	BAJA	MEDIA
2 a 5 Registros Lógicos	BAJA	MEDIA	ALTA
6 o más Registros Lógic.	MEDIA	ALTA	ALTA

Ficheros de Interfaz - EIF

Es un grupo de datos o de información de control, lógicamente relacionado, identificable por el usuario, referenciado por la aplicación, pero mantenido fuera de la frontera de la aplicación.

- Ficheros a los que accede la aplicación con el único objetivo de obtener información.
- Son mantenidos por otras aplicaciones
- Nunca los actualiza la aplicación.



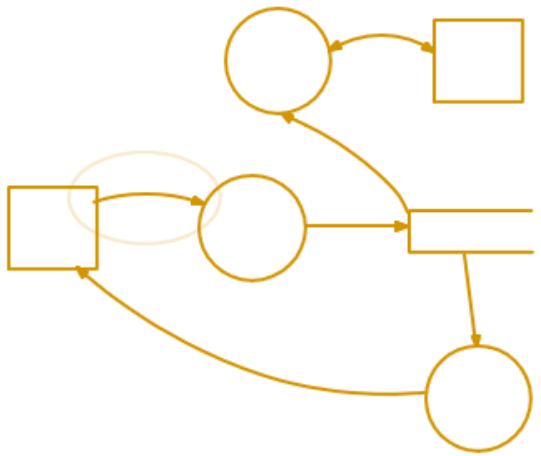
DIFICULTAD FICHEROS DE INTERFAZ	Número de Campos o Atributos		
	1-19 Atributos	20-50Atributos	51 + Atributos
1 Entidad o Registro Lógico	BAJA	BAJA	MEDIA
2 a 5 Registros Lógico	BAJA	MEDIA	ALTA
6 o más Registros Lógic.	MEDIA	ALTA	ALTA

Un EIF para una aplicación tiene que ser un ILF para alguna otra.

Entradas - EI

Es un proceso elemental en el que datos cruzan la frontera de la aplicación de afuera hacia adentro. La intención primordial es mantener uno o más ILF y/o alterar el comportamiento del sistema

- Informaciones que llegan a la aplicación desde el exterior.
- Tienen una sola dirección (Exterior a Interior)
- Siempre actualizan algún fichero interno.

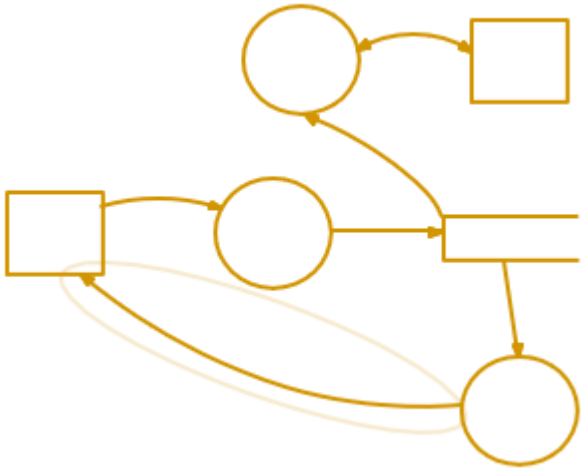


DIFICULTAD ENTRADAS	Número de Campos o Atributos de la Entrada		
	1-4 Atributos	5-15 Atributos	16 + Atributos
0 ó 1 ficheros accedidos	BAJA	BAJA	MEDIA
2 ficheros accedidos	BAJA	MEDIA	ALTA
3 + ficheros accedidos	MEDIA	ALTA	ALTA

Salidas - EO

Es un proceso elemental en el que datos derivados a partir de uno o más ILF o EOF cruzan la frontera de adentro hacia fuera. Un EO **puede** actualizar un ILF o alterar el comportamiento del sistema.

- Informaciones elaboradas por la aplicación que son transmitidas al usuario.
- Tienen una sola dirección(Interior a Exterior)

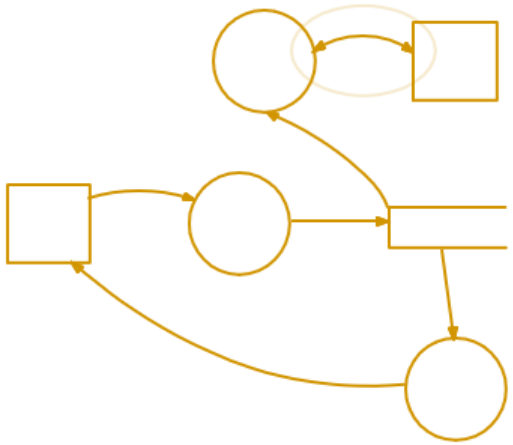


DIFICULTAD SALIDAS	Número de Campos o Atributos de la Salida		
	1-5 Atributos	6-19 Atributos	20 + Atributos
0 ó 1 ficheros accedidos	BAJA	BAJA	MEDIA
2 ó 3 ficheros accedidos	BAJA	MEDIA	ALTA
4 + ficheros accedidos	MEDIA	ALTA	ALTA

Consultas – EQ

Es un Proceso elemental en el que datos o información de control cruzan la frontera de adentro hacia fuera. **NO** incluye datos derivados y **NO** mantiene ningún ILF y **NO** altera el comportamiento del sistema

- Entradas que producen inmediatamente una salida
- No modifica los datos del sistema



- Calculamos la complejidad de la parte de entrada
- Calculamos la complejidad de la parte de salida
- Nos quedamos sólo con la complejidad **mayor** de las dos.

Hoja para calcular los Puntos de función sin ajustar (PFSA)

	Simple		Media		Compleja		Total
	Cantidad	* Peso	Cantidad	* Peso	Cantidad	* Peso	
Entradas		* 3		* 4		* 6	
Salidas		* 4		* 5		* 7	
Consultas		* 3		* 4		* 6	
Fic. Lógicos		* 7		* 10		* 15	
Fic. Interfaz		* 5		* 7		* 10	
Total puntos de función ***sin ajustar*** (PFSA)							

FACTORES DE COMPLEJIDAD

- Son catorce factores que completan la visión externo de la aplicación.
- No están recogidos en la funcionalidad de la aplicación.
- Toman un valor entre 0 y 5

Valor	Significado del valor
0	Sin influencia, factor no presente
1	Influencia insignificante, muy baja
2	Influencia moderada o baja
3	Influencia media, normal
4	Influencia alta, significativa
5	Influencia muy alta, esencial

FC1: Comunicación de datos

Los datos usados en el sistema se envían o reciben por líneas de comunicaciones.

0: Sistema aislado del exterior

1: Batch, usa periféricos E o S remotos

2: Batch, usa periféricos E y S remotos

3: Captura de datos en línea o teleproceso que pasa los datos o sistema de consulta

4: Varios teleprocesos con mismo protocolo

5: Varios protocolos. Sistema Abierto y con interfaces de todo tipo al exterior.

FC2: Proceso distribuido

Existen Procesos o Datos distribuidos, y el control de estos forma parte del sistema.

0: Sistema totalmente centralizado

1: Sistema realiza procesos en un equipo, salidas usadas vía Sw por otros equipos

2: Sistema captura, los trata en otro

3: Proceso distribuido, trans. una sola direción.

4: idem, transferencia en ambas direcciones.

5: procesos cooperantes ejecutándose en distintos equipos.

FC3: Objetivos de rendimiento

Si el rendimiento es un requisito del sistema. Es decir, es crítico algún factor como tiempo de respuesta o cantidad de operaciones por hora. Se tendrá que hacer consideraciones especiales durante el diseño, codificación y mantenimiento.

0: Rendimiento normal (no se da énfasis)

1: Se indican requisitos, no medida especial.

2: Crítico en algunos momentos. Procesos acabados antes de prox. sesión de trabajo.

3: Tiempo de respuesta es crítico.

4: en diseño hacer análisis de rendimiento en tiempo respuesta o cantidad oper./hora

5: uso herramientas para alcanzar el rendimiento demandado por el usuario

FC4: Conf. explotación usada intensamente por otros sistemas

El sistema tendrá que ejecutarse en un equipo en el que coexistirá con otros, compitiendo por los recursos, teniendo que tenerse en cuenta en la fase de diseño.

0: No se indican restricciones

1: Existen las restricciones usuales

2: Características de seguridad o tiempos.

3: Restricciones en algún procesador

4: El Sw deberá funcionar con restricciones de uso en algún procesador.

5: Restricciones especiales para aplicación en los componentes distribuidos del sistema

FC5: Tasa de transacciones

La tasa de transacciones será elevada. Se tendrá que hacer consideraciones especiales durante el diseño, codificación e instalación.

0: No se prevén picos

1: Se prevén picos poco frecuentes (mensual)

2: Se prevén picos semanales

3: Se prevén horas punta, diarias

4: Tasa de trans. tan elevada que en diseño se hace análisis de rendimiento

5: Análisis de rendimiento en diseño, implementación e instalación.

FC6: Entrada de datos en línea

La entrada de datos será directa desde el usuario a la aplicación, de forma interactiva.

0: Todo es Batch

1: 1%<entradas interactivas <7%

2: 8%<entradas interactivas <15%

3: 16%<entradas interactivas <23%

4: 24%<entradas interactivas <30%

5: Entradas interactivas >30%

FC7: Eficiencia con el usuario final

Se demanda eficiencia para el usuario en su trabajo, es decir se tiene que diseñar e implementar la aplicación con interfaces fáciles de usar y con ayudas integradas.

0: No se da énfasis al tema

1: 1 a 3 de los factores

2: 4 a 5 de los factores

3: 6 o más factores, sin requerir eficiencia

4: ... con requerimientos que implican estudio de los factores humanos en el diseño

5: … se demandan prototipos y herramientas para verificar que se alcanzaran los objetivos

FC8: Actualizaciones en línea

Los ficheros maestros y las Bases de Datos son modificadas directamente de forma interactiva.

0: No hay

1: De 1 a 3 ficheros con información de control. Cantidad baja y ficheros recuperables

2: ... pero con 4 o más ficheros de control

3: Actualización de ficheros importantes

4: ... esencial la protección ante pérdidas

5: Gran cantidad de actualizaciones interactivas. Sistemas de recuperación muy automatizados

FC9: Lógica de proceso interno compleja

- La complejidad interna en un proceso esta en función de las siguientes características:
  a) Especificados algoritmos matemáticos complejos.
  b) Proceso con lógica compleja.
  c) Especificado muchas excepciones, consecuencia de transacciones incompletas, que deberán tratarse.
  d) Manejar múltiples dispositivos de entrada/salida.
  e) Se incorporarán sistemas de seguridad y control.

0: Ninguna de las características

1: 1 Característica

2: 2 Características

3: 3 Características

4: 4 Características

5: 5 características

FC10: Reusabilidad del código

Se tendrá que hacer consideraciones especiales durante el diseño, codificación y mantenimiento para que el código se reutilice en otras aplicaciones o lugares.

Hablaremos de reutilización: Dentro de la propia aplicación, por varios sistemas, y parametrizable.

0: No se prevé

1: Reutilizar código en la misma aplicación

2: Menos de un 10% de la aplicación tiene en cuenta las necesidades de + de 1 usuario

3: El 10 % o más ...

4: Aplicación preparada para ser reutilizable. Nivel de código

5: Aplicación preparada para ser reutilizable. Por medio de parámetros

FC11: Contempla conversión e instalación

Se proveerán facilidades de conversión en el sistema, se tendrá que hacer consideraciones especiales durante el diseño, codificación y pruebas para que la conversión del sistema antiguo sea fácil de realizar durante la puesta en marcha del sistema nuevo.

0: No se requiere conversión.

1: Se solicita facilidad de instalación

2: Se solicitan procesos de conversión e instalación, no importantes para el proyecto

3: ... si son importantes

4: 2, y herramientas conversión e instalación

5: 3, y herramientas conversión e instalación. Sistema crítico para la empresa

FC12: Facilidad de operación

Operación del sistema: los trabajos asignados al centro de proceso de datos.

arranque, parada, recuperación ante fallos, copias de seguridad o minimización de las actividades manuales etc…

Se valora cuando ha sido descrita desde las primeras fases dedicándose especial atención durante el diseño, codificación y pruebas.

0: Nada, en todo caso, back-up

1 a 4: Suma de ítems

- Arranque, back-up y recuperación
- Idem, sin intervención operador (X2)
- Minimizar necesidad de dispositivos. externos almacenamiento.
- Minimiza necesidad de manejar papel

5: Sistema automático sin intervención humana

FC13: Instalaciones múltiples

El sistema ha de incluir los requerimientos de diversas empresas o departamentos en donde se ejecutará (incluso plataformas). Estas características se estarán presentes durante el diseño, codificación y pruebas.

0: 1 solo lugar

1: Múltiples lugares, mismo Hw y Sw

2: En diseño se tiene en cuenta el caso (1)

3: En diseño se tiene en cuenta múltiples entornos Hw y Sw

4: Se documenta y planea para (1) y (2)

5: Idem, para (3)

FC14: Facilidad de cambios

Se tendrá que hacer consideraciones especiales durante el diseño, codificación y mantenimiento para que en el sistema sea fácil de introducir cambios y fácil de adaptar al usuario.

Items a tener en cuenta:

- Consultas flexibles del usuario:
  - Simples con condiciones. lógicas And/Or que implican un único fichero lógico
  - Medias con condición. lógicas sobre más de 1 F.L. (X2)
  - Complejas con condiciones lógicas complejas que afectan a varios F.L. (X3)
- Parámetros de la aplicación. con tablas ajenas al código:
  - El cambio se hace efectivo al arrancar el sistema
  - El cambio es interactivo (X2)

0: No se especifica nada

1: Un ítem de valor 1

2: Items por valor 2

3: ...

5: Items por valor 5

Tabla para el calculo de los FC.

#	Factor de Complejidad	Valor (0..5)
1	*Comunicación de Datos.*	
2	*Proceso Distribuido.*	
3	*Rendimiento*	
4	*Configuración Operacional compartida*	
5	*Ratio de Transacciones*	
6	*Entrada de Datos EN-LÍNEA*	
7	*Eficiencia con el Usuario Final*	
8	*Actualizaciones EN-LÍNEA*	
9	*Complejidad del Proceso Interno*	
10	*Reusabilidad del Código*	
11	*Contempla la Conversión e Instalación*	
12	*Facilidad de Operación (back up, etc.)*	
13	*Instalaciones Múltiples*	
14	*Facilidad de Cambios*	
Factor de Complejidad Total (FCT)		↗ $Valor_i$

Calculo de los puntos de función ajustados

PFA = PFSA * (0,65 + (0.01 * FC))

- Cada factor de complejidad afecta en +/- 2,5% en los PFSA

PFSA * 65% <= PFA <= PFSA * 135%

- **Estimación del Esfuerzo Requerido**

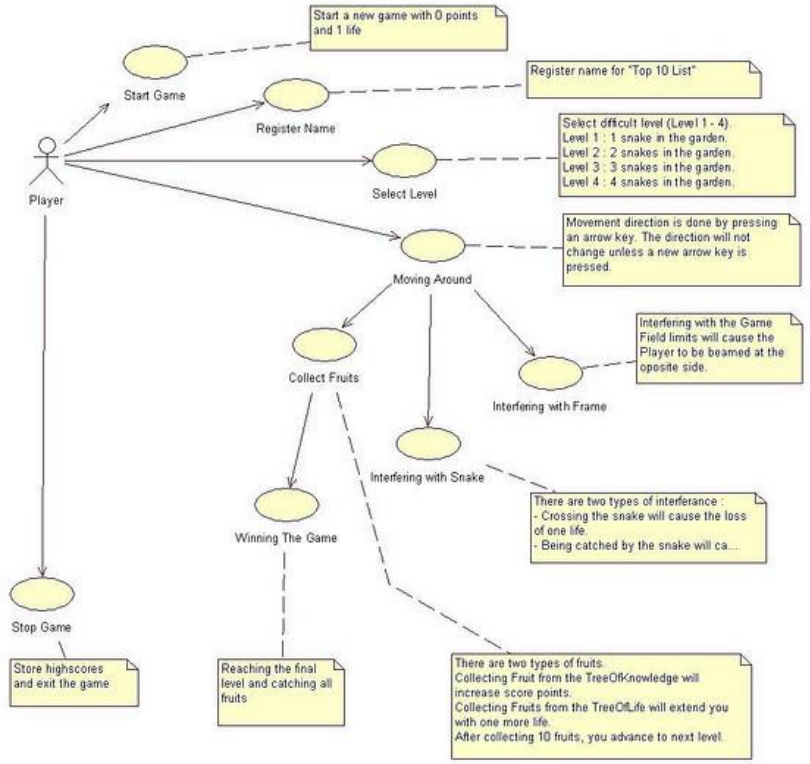
Partimos de los datos históricos de la Organización

Esfuerzo = FA * Promedio_Organización(Lenguaje)

Estimación del Esfuerzo Requerido (Datos históricos)

Nombre Proyecto	Puntos de Función	Lenguaje	Esfuerzo en horas
Sénia	200	COBOL	5.017
Paláncia	150	PASCAL	2.569
Turia	375	4GL	3.011
Albufera	500	PASCAL	9.479
Magro	425	4GL	3.342
Cabriel	800	PASCAL	13.349
Júcar	180	PASCAL	2.800
Serpis	325	4GL	2.541
Montnegre	225	PASCAL	4.528
Segura	470	COBOL	13.218

Puntos de Casos de Uso



- El método de Puntos Casos de Uso (Use Case Points) fue desarrollado en 1993 por **Gustav Kamer**, bajo la supervisión de **Ivar Jacobson** (creador de los casos de uso y gran promovedor del desarrollo de UML y el Proceso Unificado).

- Su principal ventaja es la rápida adaptación a empresas que ya estén utilizando la técnica de Casos de Uso.

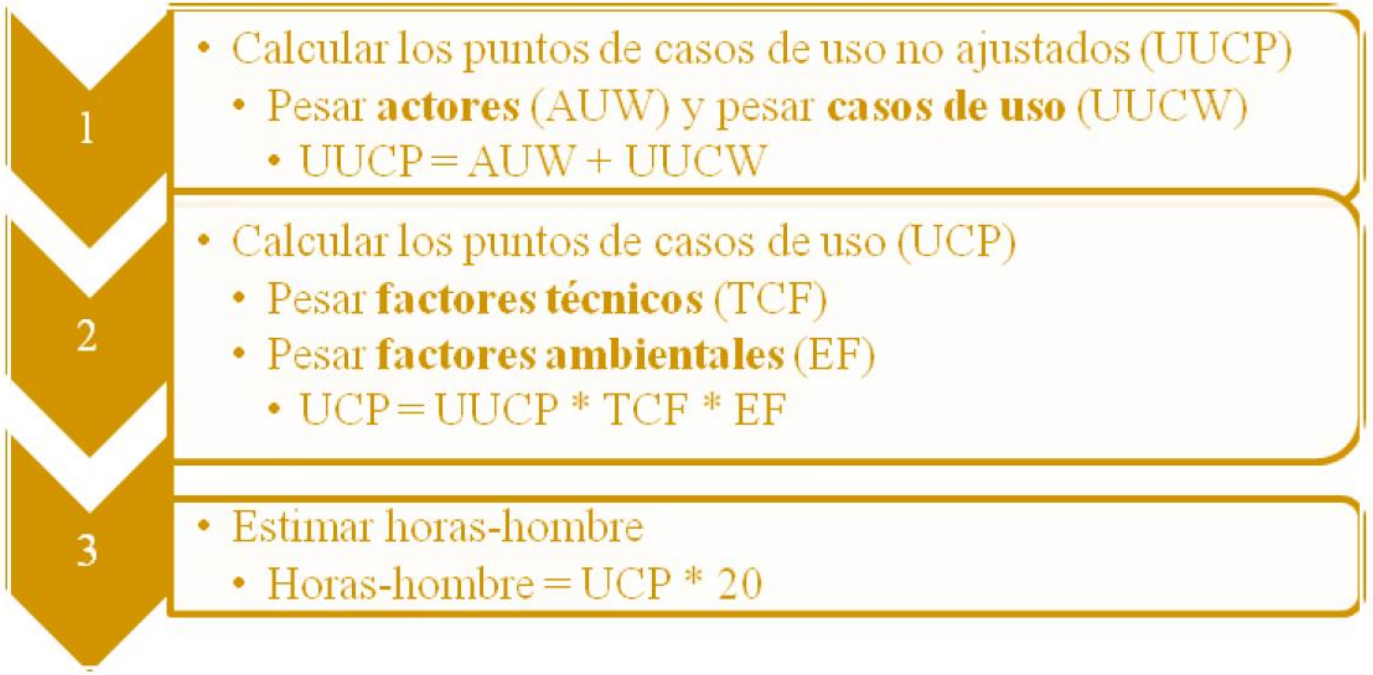
- Estima el esfuerzo (en horas-hombre) de un proyecto de desarrollo de software a partir de los casos de uso.

Ventajas – Desventajas de los PCU

Tabla 1. VENTAJAS Y DESVENTAJAS DE LA TÉCNICA DE PUNTOS DE CASOS DE USO	
Ventajas	Desventajas
Trabaja bien con diferentes tipos de software	No existe un estándar para escribir casos de uso lo que dificulta la aplicación del método.
Muestra buen rendimiento en proyectos pequeños, medianos y grandes.	Las herramientas en esta área son caras y se enfocan en la evaluación del proyecto

Vale la pena aclarar que un caso de uso por sí solo no permite efectuar una estimación de esfuerzos ni de tiempos, solamente son una herramienta para el análisis. La idea central es estimar el tamaño (cuantificar) del software a partir de los requerimientos de los casos de uso.

Pasos de la técnica de puntos de casos de uso



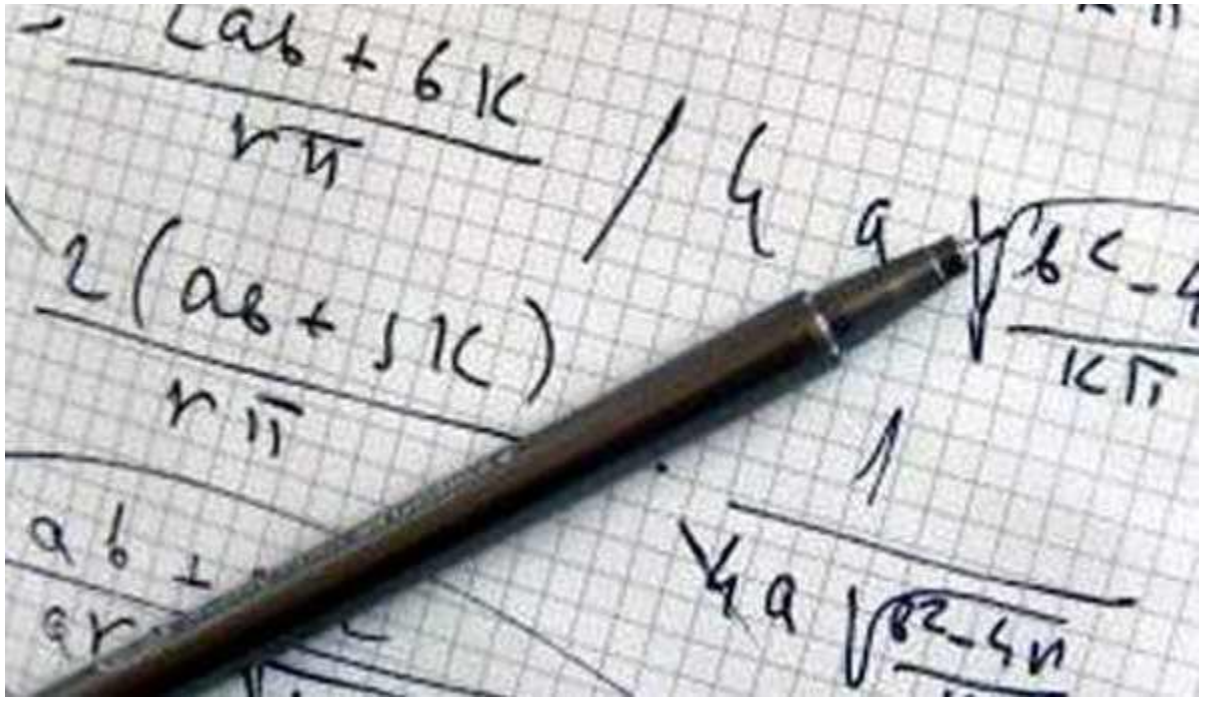
Ecuación UCP (Use Case Points)

- **UCP=UUCP*TCF*ECF**

- Donde:
- UUCP = Puntos de Caso de Uso sin ajustar (Unadjusted Use Case Points)
- TCF = Factor de complejidad técnica (Technical Complexity Factor)
- ECF = Factor de Complejidad del Medio Ambiente (Environment Complexity Factor)
- PF = Factor de productividad (Productivity Factor)

- **NOTA:** Cuando la productividad se incluye como un coeficiente que expresa el tiempo, entonces la ecuación puede utilizarse para estimar el número de horas-hombre necesarias para completar un proyecto.

Ecuación UCP (Use Case Points)



- Hay que hacer algunos cálculos

UUCP (Puntos de Caso de Uso sin ajustar)

- Da una idea un poco más precisa de la dificultad de los casos de uso e interfaces

- **UUCP = UAW + UUCW**

- Se calculan con base en:
- Pesos de los Casos de Uso sin Ajustar (**UUCW**): Basado en el número total de actividades (o pasos) contenidos en todos los escenarios del caso de uso.
- Pesos de los Actores sin Ajustar (**UAW**) : Basado en la combinación de la complejidad de todos los actores en todos los casos de uso.

UUCW – Peso de Actores

Tabla 2. PESO DE LOS ACTORES		
Tipo de actor	Descripción	Factor
Simple	Otro sistema con una API definida	1
Medio	Otro sistema interactuando con algún protocolo (TCP) o una persona interactuando a través de una interfaz en modo texto	2
Complejo	Una persona interactuando a través de una interfaz gráfica de usuario	3

UUCW – Peso de CUS

Tabla 3. PESO DE LOS CASOS DE USO		
Tipo de caso de uso	Descripción	Factor
Simple	3 transacciones o menos	5
Medio	4 a 7 transacciones	10
Complejo	Más de 7 transacciones	15

Simple : *Interfaz de usuario simple*. Toca solo una única entidad de la Base de datos. Su escenario de éxito tiene 3 pasos o menos. Su implementación involucra menos de 5 clases

Medio : *Más diseño de interfaz*. Toca 2 o más entidades de base de datos. Entre 4 y 7 pasos. Su implementación implica entre cinco y 10 clases

Complejo: *Interfaz de usuario compleja o procesamiento.* Toca 3 o más entidades de la Base de datos. Más de 7 pasos. Su implementación implica más de 10 clases.

UUCW

Luego, se debe completar este cuadro:

Tipo de Caso de Uso	Descripción	Peso (factor)	Número de Casos de Uso	Resultado
Simple	Transacciones= 3 ó menos Clases= Menos de 5	5		
Medio	Transacciones= 4 a 7 Clases= 5 a 10	10		
Complejo	Transacciones= Más de 7 transacciones Clases= Más de 10 clases	15		
			Total UUCW	

Resultado = Factor x número de CUs

Total es la sumatoria de Resultados

UAW

- Consiste en la evaluación de la complejidad de los actores con los que tendrá que interactuar el sistema

Tabla 2. PESO DE LOS ACTORES		
Tipo de actor	Descripción	Factor
Simple	Otro sistema con una API definida	1
Medio	Otro sistema interactuando con algún protocolo (TCP) o una persona interactuando a través de una interfaz en modo texto	2
Complejo	Una persona interactuando a través de una interfaz gráfica de usuario	3

Luego, se debe completar este cuadro:

Tipo de Actor	Descripción	Peso (factor)	Número de Actores	Resultado
Simple	Otro sistema que interactúa con el sistema a desarrollar mediante una interfaz de programación (API).	1		
Medio	Otro sistema interactuando a través de un protocolo (ej. TCP/IP) o una persona interactuando a través de una interfaz en modo texto.	2		
Complejo	Una persona que interactúa con el sistema mediante una interfaz gráfica (GUI).	3		
			Total UAW	

Resultado = Factor x número de actores

Total es la sumatoria de Resultados

TCF (Factor de Complejidad Técnica)

- **Son 13 puntos** que evalúan la complejidad de los módulos del sistema que se desarrolla. Cada uno tiene un peso predefinido y un factor subjetivo percibido por el equipo de desarrollo.

Tabla 4. FACTORES TÉCNICOS		
Factor	Descripción	Peso
T1	Sistema distribuido	2
T2	Objetivos de performance o tiempo de respuesta	1
T3	Eficiencia del usuario final	1
T4	Procesamiento interno complejo	1
T5	El código debe ser reutilizable	1
T6	Facilidad de instalación	0.5
T7	Facilidad de uso	0.5
T8	Portabilidad	2
T9	Facilidad de cambio	1
T10	Concurrencia	1
T11	Objetivos especiales de seguridad	1
T12	Acceso directo a terceras partes	1
T13	Facilidades especiales de entrenamiento a usuarios	1

Tabla 5. ESCALAS DE ESTIMACION TCF	
Descripción	Valor
Irrelevante	De 0 a 2
Medio	De 3 a 4
Esencial	5

Cada uno de estos factores se deben evaluar según la siguiente escala SUBJETIVA:

Descripción	Valor
Irrelevante	0 a 2
Medio	3 a 4
Esencial	5

UNIVERSIDAD DE LIMA

60 años

TCF (Factor de Complejidad Técnica)

- Luego, se debe completar este cuadro:

Factor técnico	Descripción	Peso	Impacto percibido	Factor calculado
T1	Sistema distribuido	2		
T2	Rendimiento o tiempo de respuesta	1		
T3	Eficiencia del usuario final	1		
T4	Procesamiento interno complejo	1		
T5	El código debe ser reutilizable	1		
T6	Facilidad de instalación	0.5		
T7	Facilidad de uso	0.5		
T8	Portabilidad	2		
T9	Facilidad de cambio	1		
T10	Concurrencia	1		
T11	Características especiales de seguridad	1		
T12	Provee acceso directo a terceras partes	1		
T13	Se requiere facilidades especiales de entrenamiento a usuario	1		
			Factor Total Técnico	

Resultado = Peso x impacto

Total es la sumatoria de Resultados

TCF = 0.6 + (0.01* Factor Total Técnico)

UNIVERSIDAD DE LIMA

60 años

ECF (Factor de Complejidad Ambiental)

- Establece la experiencia del equipo de desarrollo: están relacionados con las habilidades y experiencia del grupo de personas involucradas con el desarrollo del proyecto.

Tabla 6. FACTORES AMBIENTALES

Factor	Descripción	Peso
E1	Familiaridad con el modelo del proyecto utilizado	1.5
E2	Experiencia en la aplicación	0.5
E3	Experiencia en orientación a objetos	1
E4	Capacidad del analista líder	0.5
E5	Motivación	1
E6	Estabilidad en los requerimientos	2
E7	Personal de medio tiempo	-1
E8	Dificultad en el lenguaje de programación	-1

Tabla 7. ESCALAS DE ESTIMACION EF

Descripción	Valor
Sin experiencia, sin motivación, estabilidad	De 0 a 2
Promedio	3
Amplia experiencia, motivación, estabilidad	De 3 a 5

Cada uno de estos factores se deben evaluar de **0 a 5** según la siguiente escala SUBJETIVA:

- **Un valor de 1** significa que el factor tiene un fuerte impacto negativo para el proyecto,
- **3** es medio
- **5** significa que tiene un fuerte impacto positivo

ECF (Factor de Complejidad Ambiental)

Factor Ambien-tal	Descripción	Peso	Impacto perci-bido	Factor calculado
E1	Familiaridad con el modelo de proyecto utilizado Familiaridad con UML	1.5		
E2	Personal tiempo parcial	-1		
E3	Capacidad del analista líder	0.5		
E4	Experiencia en la aplicación	0.5		
E5	Experiencia en orientación a objetos	1		
E6	Motivación	1		
E7	Dificultad del lenguaje de programación	-1		
E8	Estabilidad de los requerimientos	2		
11/04/2011			Factor Ambiental Total	

Resultado = Peso x impacto

Total es la sumatoria de Resultados

ECF = 1.4 + (-0.03* Factor Total Ambiental)

PF (Factor de Productividad)

- El PF es la **proporción de horas hombre de desarrollo necesario por caso de uso**. Estadísticas de proyectos pasados proporcionan los datos para estimar la PF inicial.
- Por ejemplo, si un proyecto pasado con un UCP de 120 tomó 2,200 horas para completarse, divida 2,200 por 120 para obtener un PF de 18 horas hombre por punto de caso de uso.

PF (Factor de Productividad)

- El método original de Karner propone usar un factor de ajuste de 20 personas-hora por cada Punto Caso de Uso (UCP).

Horas-hombre sistema de procesamiento de órdenes = 57.77 * 20 = 1155.52

~ 29 semanas a 40 horas por semana, para una persona

Tomando un equipo pequeño de 6 personas trabajando full-time

~ 5 semanas de esfuerzo

- Barnerjee propone un rango entre 15 y 30 horas

PF (Factor de Productividad)

Factor Ambiental	Descripción del factor	Peso
E1	Familiaridad con el modelo de proyecto	1.5
E2	Experiencia en la aplicación	0.5
E3	Experiencia en orientación a objetos	1
E4	Capacidad del analista líder	0.5
E5	Motivación	1
E6	Estabilidad de los requerimientos	2
E7	Personal part-time	-1
E8	Dificultad del lenguaje de programación	-1

PF (Factor de Productividad)

- Zcheider y Winters sugiere un refinamiento de los factores de entorno (EF) y seguir el procedimiento que se presenta a continuación:
  - Contar cuántos factores ambientales desde E1 a E6 son inferiores a 3 (valor de nivel promedio) y cuántos de los factores ambientales E7 y E8 son superiores a 3
  - Entonces se usan
    - **20** horas-hombre por UCP **si el valor es ≤ 2**
    - **28** horas-hombre por UCP **si el valor es ≤ 4**
    - **36** horas-hombre por UCP **si el valor es ≥ 5**
      - En este caso considerar modificar el proyecto ya que es muy riesgoso

Esfuerzo = UCP * Factor de Productividad

PF (Factor de Productividad)

Es inferior a 3

Factor Ambiental	Peso	ELevel	ELevel * Peso	Justificación
E1	1.5	1	1.5	La mayoría del equipo no familiarizado
E2	0.5	3	1.5	La mayoría del equipo son programadores
E3	1	3	3	Programadores OO
E4	0.5	5	2.5	El líder es idóneo
E5	1	5	5	El equipo está entusiasmado
E6	2	5	10	No se esperan cambios
E7	-1	0	0	Sin empleados a tiempo parcial
E8	-1	3	-3	Se programará con Java
Tenemos un EF inferior al promedio			**=>**	**20 horas-hombre por UCP**

Estimación del proyecto

Estimar Horas-hombre Sistema Procesamiento de órdenes
Para un **UCP= 57.77**
Horas-hombre sistema de procesamiento de órdenes = 57.77 * 20 = 1155.52

Actividad	Porcentaje
Análisis	10%
Diseño	20%
Programación	40%
Pruebas	15%
Sobrecarga (otras actividades)	15%

Actividad	Porcentaje	Horas-hombre
Análisis	10%	288.88
Diseño	20%	577.76
Programación	**40%**	**1155.52**
Pruebas	15%	433.32
Sobrecarga (otras actividades)	15%	433.32
TOTAL Esfuerzo	**100%**	**2888.8**

~ 72 semanas a 40 horas por semana, para una persona
Tomando un equipo de 6 personas trabajando full-time:
~ 12 semanas de esfuerzo

Horas Estimadas

Total horas estimadas = UCP*PF

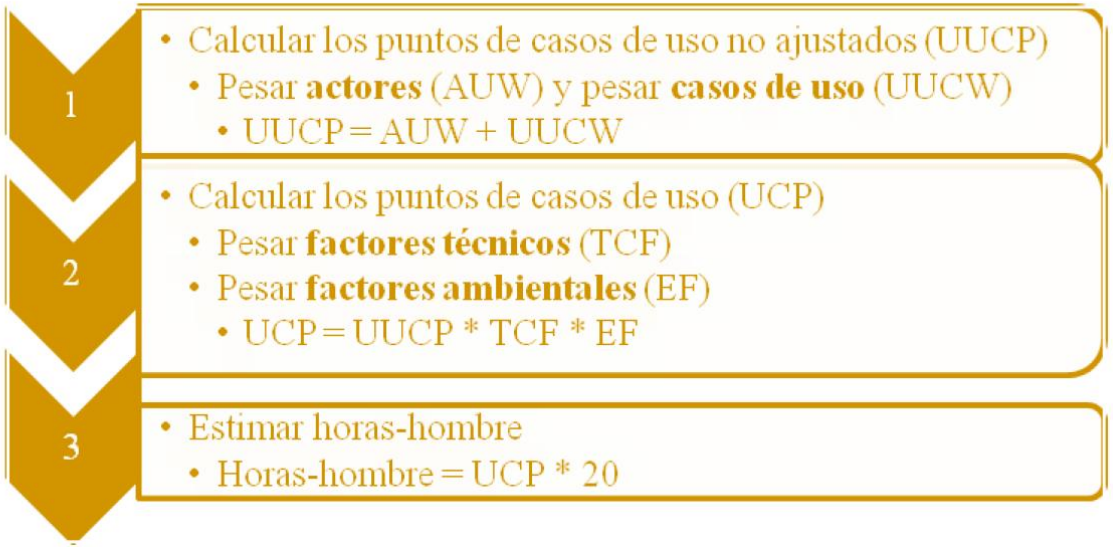
Si no hay datos históricos que hayan sido recabados se toman en cuenta dos posibilidades:

1. Establecer una base para cálculos de los UCP de proyectos completados anteriormente.
2. Utilice un valor entre **15 y 30** dependiendo de la experiencia y logros pasados del equipo de desarrollo. Si se trata de un nuevo equipo, use un valor de **20** para el primer proyecto.

Horas Estimadas

Siguiente paso de esta técnica nos indica que debemos estimar las horas hombre mediante la fórmula:

HH = UCP * 20



Horas Estimadas

El autor de la técnica sugiere usar 20 horas hombre por UCP. Por ejemplo, para un sistema de 60 UCP * 20 hrs/hombre nos da un total de 1200 hrs/hombre. Lo que equivale a 30 semanas (40 hrs por semana para una persona), de esta forma, un equipo de 5 personas desarrollarían el sistema en 6 semanas. En resumen:

Tabla 8. RESUMEN EJEMPLO

Actividad	Porcentaje de tiempo	Horas-Hombre
Análisis	10%	300
Diseño	20%	600
Programación	40%	1200
Pruebas	15%	450
Sobrecarga	15%	450
TOTAL	100%	3000

Ejemplo

Para demostrar los pasos en los que se basa la técnica, realizaremos un caso práctico para el caso de uso **"retirar dinero de cajero automático"**. El proceso se explica con el fin de lograr una mayor comprensión de la técnica y los pasos asociados.

- ***Actor:*** cliente.
- ***Propósito:*** realizar un retiro de una cuenta desde un cajero automático.
- ***Visión general****:* un cliente llega a un cajero automático, introduce la tarjeta, se identifica y solicita realizar un retiro de dinero. El cajero le da el dinero solicitado tras comprobar que la operación puede realizarse. El cliente toma el dinero y se va.

Ejemplo

Curso típico de eventos :

1. Este caso de uso empieza cuando un cliente introduce su tarjeta en el cajero.
2. El sistema pide la clave de identificación.
3. El cliente introduce la clave.
4. El sistema presenta las opciones disponibles.
5. El cliente selecciona la operación de Retiro.
6. El sistema pide la cantidad a retirar.
7. El cliente introduce la cantidad requerida.
8. El sistema procesa la petición y, eventualmente, da el dinero solicitado.
9. El sistema devuelve la tarjeta y genera un recibo.
10. El cliente recoge el dinero, el recibo y la tarjeta. Procede a retirarse

Ejemplo

calcular los puntos de casos de uso no ajustados (UUCP):

PESO ACTORES EJEMPLO		
Actores	Tipo	Valor
Cliente	Complejo	3
UAW		3

PESO CASOS DE USO EJEMPLO		
Caso de uso	Tipo	Valor
Retirar dinero cajero	Promedio	10
UUCW		10

$$\textbf{UUCP} = \text{UAW} + \text{UUCW} = 3 + 10 = \textbf{13}$$

Ejemplo

calcularemos los puntos de caso de uso ajustados (UCP)

FACTORES TÉCNICOS EJEMPLO				
Factor	Descripción	Peso	Nivel	Pes*Niv
T1	Sistema distribuido	2	3	6
T2	Performance o tiempos de respuesta	1	5	5
T3	Eficiencia del usuario final	1	1	1
T4	Procesamiento interno complejo	1	1	1
T5	Código reutilizable	1	0	0
T6	Facilidad de instalación	0.5	1	0.5
T7	Facilidad de uso	0.5	2	1
T8	Portabilidad	2	0	0
T9	Facilidad de cambio	1	1	1
T10	Concurrencia	1	5	5
T11	Seguridad	1	5	5
T12	Acceso directo a terceras partes	1	0	0
T13	Facilidades especiales	1	0	0
		TFactor = ∑ (nivel * peso)		25.5

El peso de los factores técnicos será:

TCF = 0.6 + (0.01 * *TFactor*) = 0.6 + (0.01 * 25.5) = **0.855**

Ejemplo:

FACTORES AMBIENTALES EJEMPLO				
Factor	Descripción	Peso	Nivel	Pes*Niv
E1	Familiaridad con el modelo	1.5	2	3
E2	Experiencia en la aplicación	0.5	3	1.5
E3	Experiencia en orientación a objetos	1	5	5
E4	Capacidad del analista líder	0.5	5	2.5
E5	Motivación	1	5	5
E6	Estabilidad en los requerimientos	2	5	10
E7	Personal de medio tiempo	-1	5	-5
E8	Dificultad en el lenguaje de programación	-1	0	0
		EFactor = ∑(Peso * Nivel)		22

El peso de los factores ambientales lo calculamos de acuerdo a la fórmula propuesta:

EF = 1.4 + (-0.03 * *EFactor*) = 1.4 + (-0.03 * 22) = **0.74**

Ejemplo :

Los resultados finales los mostramos en la siguiente tabla:

HORAS-HOMBRE TOTALES EJEMPLO		
Actividad	**Porcentaje**	**Horas Hombre**
Análisis	**10%**	**41.12**
Diseño	**20%**	**82.25**
Programación	**40%**	**164.50**
Pruebas	**15%**	**61.68**
Sobrecarga	**15%**	**61.68**
Total esfuerzo	**100%**	**411.25**

- Analizando la tabla, podemos ver que el proceso total de desarrollo del caso de uso de este ejemplo son **411.25 hrs**, el equivalente a **10.28 semanas** para una sola persona (trabajando tiempo completo a 40 hrs por semana).
- Para un equipo de desarrollo de **5 personas** con trabajo de tiempo completo, demoraría **2.05 semanas** en crearse un sistema con dichas características.
- Los costos de producción de sistema se calcular en función del número de horas trabajadas, multiplicando el costo deseado por hora y el total de ellas. Por ejemplo, si el costo por hora es de $300, el costo total del sistema seria:

  Total = 411.25 hrs * 300 = **$123,376.5**
- Y el plazo de entrega seria de 2 semanas, con equipo de trabajo de 5 personas de tiempo completo.

Los tiempos estimados por etapa (análisis, diseño, programación, etc.), así como los costos por hora, son criterio del equipo de desarrollo o de la empresa encargada de crear el sistema y dependen en gran medida de la experiencia de ellos.

Ejemplo :

Los puntos de casos de uso (UCP) para el caso de uso “retirar dinero de cajero automático” son:

$$\mathbf{UCP} = UUCP * TCF * EF = 13 * 0.855 * 0.74 = \mathbf{8.225}$$

- Tomando en consideración la propuesta del creador de la técnica descrita en este trabajo, asignamos **20 hrs-hombre por punto de casos de uso**,
- dando un total de ***164.50 Hrs-Hombre***,
- es decir, 4.11 semanas para una sola persona trabajando tiempo completo.

COCOMO

COCOMO II permite realizar estimaciones en función del tamaño del software, y de un conjunto de factores de coste y de escala.El modelo constructivo de coste (COCOMO) es una jerarquía de modelos de estimación

Modelo 1. El modelo básico es univariable estático que calcula el esfuerzo del desarrollo del software en función del tamaño del programa, expresado en LDC estimadas.

Modelo 2. El modelo intermedio calcula el esfuerzo de desarrollo en función del tamaño del programa y de un conjunto de "conductores de coste", que incluye la evaluación subjetiva del producto, del hardware, del personal y de los atributos del proyecto.

Modelo 3. El modelo avanzado incorpora todas las características de la versión intermedia y lleva a cabo una evaluación del impacto de los "conductores de coste" en cada fase (análisis, diseño, etc.) del proceso de ingeniería del software.

COCOMO

Los modelos anteriores están definidos para tres tipos de proyectos, definidos por Boehm:

- **Modo orgánico**. Proyectos de software relativamente pequeños y sencillos en los que trabajan pequeños equipos, con buena experiencia en las aplicaciones, sobre un conjunto de requisitos poco rígidos.

- **Modo semiacoplado**. Proyectos de software **intermedios** (en tamaño y complejidad) en los que los equipos, con variados niveles de experiencia deben satisfacer los requisitos poco o medio-rígidos (p.e. Un sistema de procesamiento de transacciones con requisitos fijos para un hardware de terminal o un software de gestión de base de datos).

- **Modo empotrado**. Proyectos de software que deben ser desarrollados en un conjunto de hardware, software y restricciones operativas muy restringido (p.e. software de control de navegación para un avión).

COCOMO. Modelo básico-ejemplo.

- Aplicando el modelo COCOMO al ejemplo de software CAD.
- Las LDC estimadas son 33360.
- Coeficientes de la tabla COCOMO básico para el modelo semiacoplado
  - $E = 3,0\ (33,33)^{1,12} = 1523$ personas-mes
  - $D = 2,5\ (152)^{0,35} = 14,5$ meses
- El valor de la duración del proyecto permite al planificador recomendar un número de N personas para el proyecto:
  - $N = E / D = 11$ personas

COCOMO. Modelo básico.

- Las ecuaciones del **modelo básico** son las siguientes:
  - $PM=E = a\ (KLDC)^b$
  - $TDEV=D = c\ (E)^d$

Modo de Desarrollo	Esfuerzo	Cronograma
Orgánico	$PM=2.4 \times (KSLOC)^{1.05}$	$TDEV=2.5 \times (PM)^{0.38}$
Semiacoplado	$PM=3.0 \times (KSLOC)^{1.12}$	$TDEV=2.5 \times (PM)^{0.35}$
Empotrado	$PM=3.6 \times (KSLOC)^{1.20}$	$TDEV=2.5 \times (PM)^{0.32}$

Tabla 1: Ecuaciones del Modelo Básico de COCOMO 81. [Boehm 1981]

  - **E es el esfuerzo** aplicado en persona-mes.
  - **D es el tiempo** de desarrollo en meses.
  - KLDC es el número estimado de líneas de código.
  - a, b, c y d se muestran en la siguiente tabla

Tabla 3.1. COCOMO básico

Proyecto de software	a_b	b_b	c_b	d_b
Orgánico	2,4	1,05	2,5	0,38
Semiacoplado	3,0	1,12	2,5	0,35
Empotrado	3,6	1,20	2,5	0,32

COCOMO. Modelo Intermedio.

- El modelo básico se amplia para considerar un conjunto de **atributos conductores de coste**:

1. Atributos del producto
    a. Fiabilidad del software requerida.
    b. Tamaño de la base de datos de la aplicación.
    c. Complejidad del proyecto.

2. Atributos del hardware
    a. Restricciones de rendimiento en tiempo de ejecución.
    b. Restricciones de memoria.
    c. Volatilidad del entorno de la máquina virtual.
    d. Tiempo de espera requerido.

3. Atributo del personal
    a. Capacidad de análisis.
    b. Capacidad del ingeniero de software.
    c. Experiencia en aplicaciones.
    d. Experiencia con la máquina virtual.
    e. Experiencia con el lenguaje de programación.

4. Atributos del proyecto
    a. Utilización de herramientas de software.
    b. Aplicaciones de métodos de ingeniería del software.
    c. Planificación temporal del desarrollo requerida.

COCOMO. Modelo Intermedio.

- Cada uno de los atributos es valorado de 0 a 6 puntos.
- De acuerdo con la evaluación se determina un multiplicador de esfuerzo a partir de las tablas calculadas por Boehm.
- Con el producto de todos los multiplicadores de esfuerzo se obtiene un **factor de ajuste del esfuerzo** (FAE), que toma valores típicos entre 0,9 y 1,4.

- La ecuación del modelo COCOMO intermedio es:

  - $E = a\ (KLDC)^b\ FAE$
    - E es el esfuerzo en personas-mes.
    - LDC es el número estimado de líneas de código.
    - a y b se obtienen de la tabla:

- COCOMO es el modelo empírico más completo para la estimación del software publicado hasta la fecha. Deben tenerse en cuenta los propios comentarios de Boehm sobre COCOMO y sobre todos los modelos: Un modelo de estimación de costes del software está bien construido si puede estimar los costes del desarrollo de software en torno al 20 por 100 de los costes reales, y el 70 por 100 del tiempo y ello en su propio terreno.