IIC2143: Ingeniería de Software

Estimaciones en proyectos de desarrollo de software

2-2018

Yadran Eterovic (yadran@ing.puc.cl)

Porque un hombre bien instruido es uno que busca aquel grado de precisión en cada tipo de estudio que la naturaleza del asunto [a la mano] permite.

—Aristóteles, 330 AC

Cuidado con el significado de *estimación*

Según el Diccionario de la Lengua Española de la RAE (23a ed.), **estimación** es

> *"1. Aprecio y valor que se da y en que se tasa y considera algo.*
>
> *2. Aprecio, consideración, afecto."*

¿Qué significa *estimate* (en inglés)?

Según el diccionario del inglés de mi computador, **estimate** es [traducción mía]

> *"Un juicio o cálculo aproximado del valor, número, cantidad o extensión de algo"*

Este es el significado de la palabra "estimación" en "Estimaciones en proyectos de desarrollo de software"

Distingamos entre *estimaciones*, *planes, metas* y *compromisos*

Pero cuando nos piden una *estimación* para un proyecto de software,

… ¿nos están pidiendo un cálculo aproximado, tentativo o preliminar?

No:

- normalmente, nos están pidiendo un *compromiso* o un *plan* para lograr una *meta*
- no se espera que cambiemos de opinión más adelante

Una **meta** es una afirmación de un objetivo de negocio deseable (pero no por eso alcanzable)

“Necesitamos tener la versión 2 lista para el próximo proceso de matrícula”

“Debemos limitar el costo del próximo release a UF10,000”

Un **compromiso** es una promesa de entregar una determinada funcionalidad

… con un nivel de calidad específico

… en una cierta fecha

(… siempre y cuando se cumplan ciertas condiciones)

Un compromiso no tiene por qué ser igual que la estimación

Hay una diferencia esencial entre *estimación* y *planificación*

Una *estimación* debiera ser un **proceso analítico no sesgado**

Una *planificación* es un **proceso sesgado** en pos del logro de metas

Una estimación debiera ser un **proceso analítico no sesgado**

Es peligroso querer que una estimación “salga” (resulte) de una cierta manera

El objetivo es exactitud, corrección

… y no buscar un resultado particular

Las estimaciones son los fundamentos de los planes …

Una planificación es **un proceso sesgado en pos del logro de metas**

El objetivo de la planificación es conseguir un resultado particular

Sesgamos los planes —planificamos medios específicos— para lograr resultados específicos

… pero los planes no tienen por qué ser lo mismo que las estimaciones:

si las estimaciones son muy diferentes de las metas,

… entonces los planes debieran reconocer la brecha y considerar un alto nivel de riesgo

No hay que mezclar *estimaciones* con *planificación*

Como hay diferencias fundamentales entre ambos,

… mezclarlos lleva a malas estimaciones y malos planes

¡Cuidado! Si hay un objetivo de planificación muy importante,

… los participantes del proyecto podrían referirse al objetivo como una “estimación”,

… dándole un halo de objetividad que no tiene

Ejemplos de consideraciones de planificación que dependen de buenas estimaciones

Crear un calendario de trabajo detallado

Identificar la ruta crítica de un proyecto

Priorizar las funcionalidades para las entregas

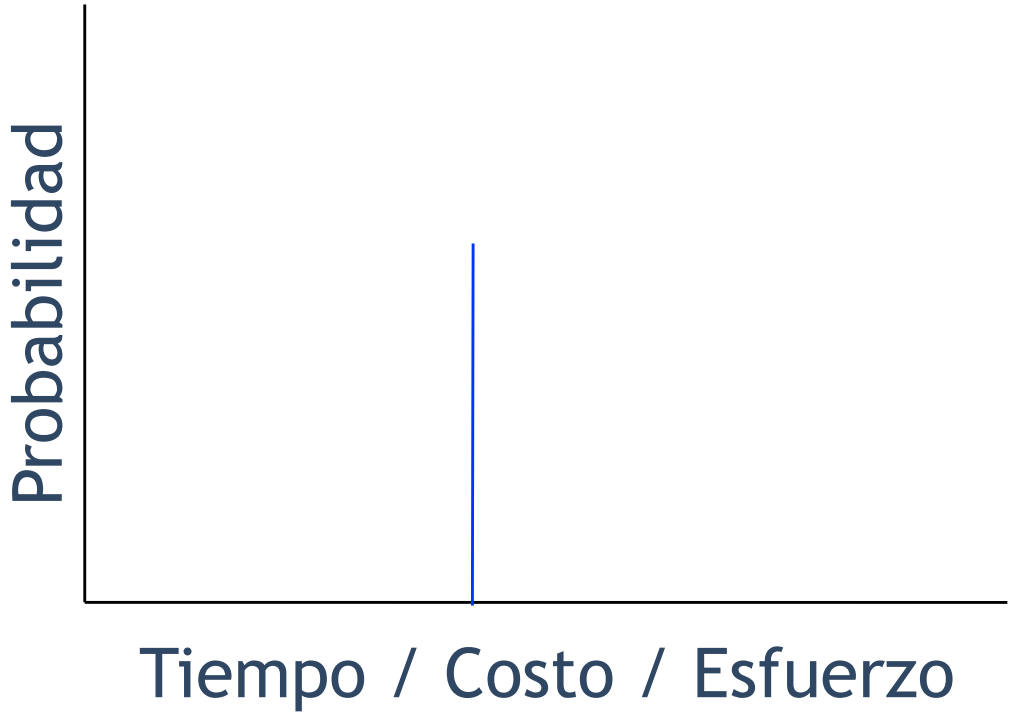
Dividir un proyecto en iteraciones

En resumen

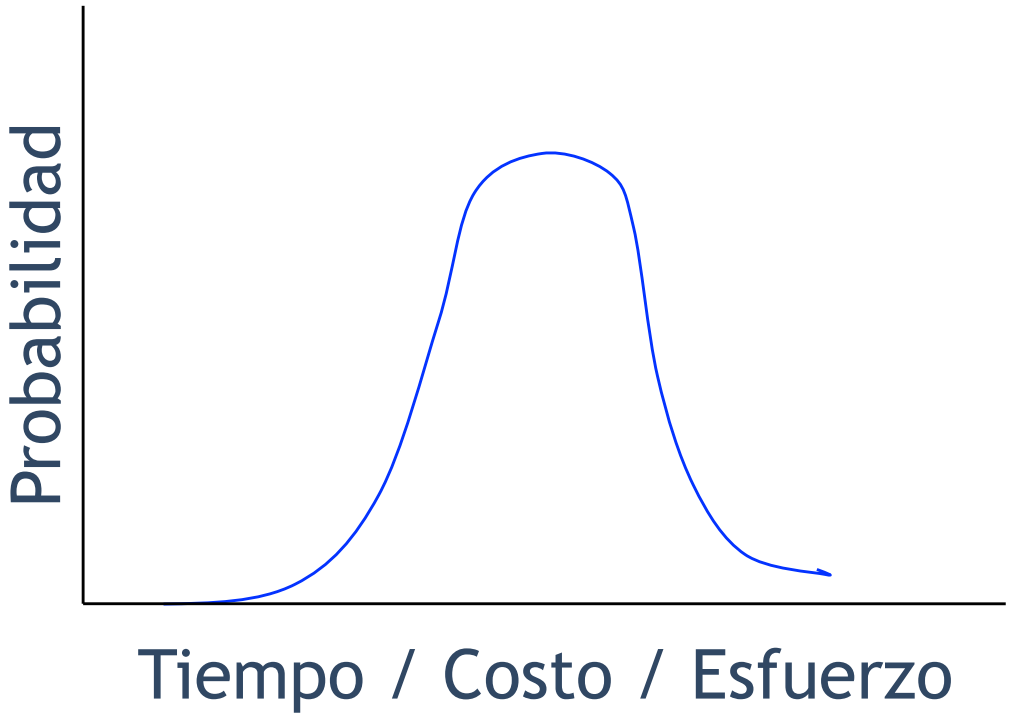
Cuando te pidan dar una estimación,

… aclara si realmente se supone que hagas una estimación

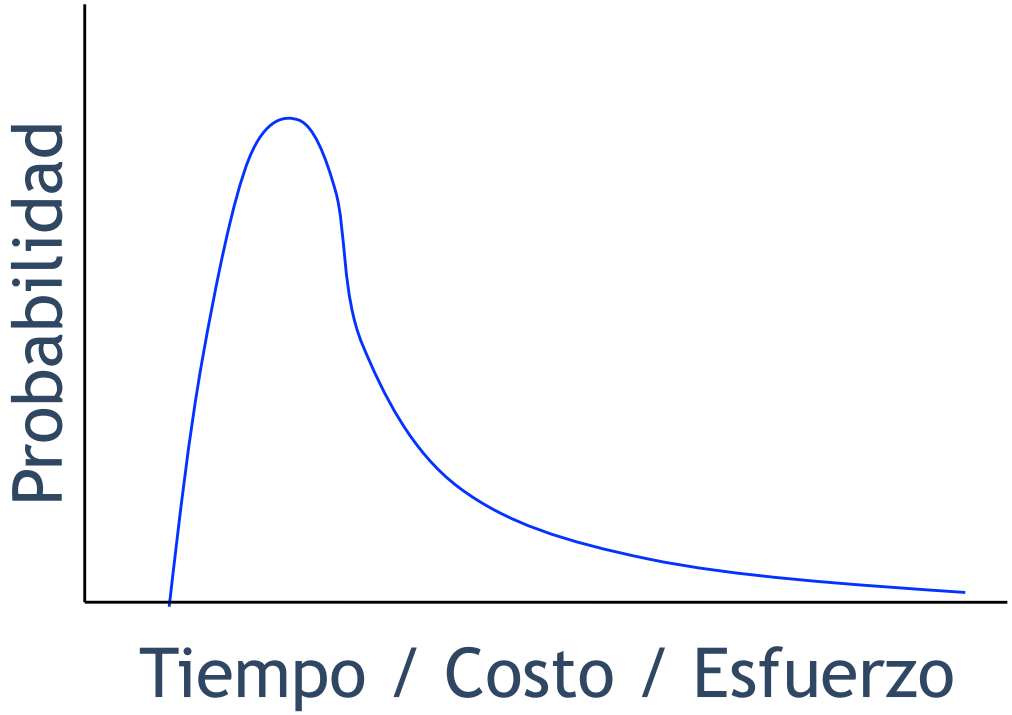
… o si se supone que hagas un plan para conseguir una meta



¿Es una estimación o
es realmente una meta?
¿Cuál es su probabilidad?



La suposición de que los resultados de un proyecto de software siguen una curva normal, con forma de campana, es incorrecta



Resultados posibles de un proyecto de software:
hay un límite para qué tan bien puede resultar,
pero no lo hay para qué tan mal.

¿Cuál es el estado de la práctica de las estimaciones de software?

Hay organizaciones sofisticadas que logran resultados para sus proyectos que están dentro del ±10% de los resultados estimados:

> para estas organizaciones es muy importante la investigación que se hace en el tema, que les permitiría lograr resultados dentro del ±5%

Pero las organizaciones típicas preocupadas por el tema luchan para evitar estimaciones que estén equivocadas en un 100% o más

Es más fácil estimar proyectos pequeños que proyectos grandes

Tamaño (FP)	A tiempo (%)	Tarde (%)	Cancelado (%)
10	92	6	2
100	81	12	7
1,000	62	18	20
10,000	28	24	48
100,000	14	21	65

Si 1FP ≈ 100 líneas, el rango de tamaños va desde mil líneas hasta 10 millones de líneas

¿Qué es un **modelo de estimación**?

Es un modelo para ayudarnos a razonar sobre las implicaciones en cuanto a costo y tiempo de las decisiones de software que necesitamos tomar

… decisiones como las de la próxima diapositiva

Hacer inversiones que incluyen desarrollo de software

Establecer presupuestos y plazos para planificar y controlar un proyecto

Decidir acerca de, o negociar, compromisos entre costo, plazos, funcionalidad, desempeño o calidad del software

Tomar decisiones de gestión de riesgos sobre costos y plazos

Decidir qué partes de un sistema desarrollar, reusar, arrendar o comprar

Decidir sobre el inventario de software legado: qué partes modificar, sacar de servicio, subcontratar

Establecer estrategias de inversión mixtas para mejorar las capacidades de software de la organización

Decidir cómo implementar una estrategia de mejora-miento de procesos

¿Qué es una buena estimación?

(Definición común de “buena estimación”:

“La predicción más optimista que puede llegar a hacerse realidad” **)**

Ahora sí: ¿Qué es una *buena estimación*?

Es una estimación que entrega una visión suficientemente clara de la realidad del proyecto,

… tal que permite que la gerencia tome buenas decisiones acerca de cómo controlar el proyecto

… para que éste alcance sus metas

Una buena estimación depende de **conocer bien** tres factores principales

La *magnitud* del software

El *tipo* de software

El *equipo de desarrollo* de software

La **magnitud del software** es el factor principal

… es el contribuyente individual más significativo al esfuerzo y a la duración del proyecto:

invirtamos una cantidad apropiada de esfuerzo para evaluar la magnitud del software que se va a desarrollar —en sí, una estimación

El **tipo de software** también contribuye al esfuerzo y a la duración del proyecto

Si estamos desarrollando software crítico (vida de personas, tiempo real),

… entonces el proyecto va a requirir mucho más esfuerzo que un proyecto de tamaño similar de sistemas de negocio

Incluyamos el tipo de software en nuestras estimaciones:

- es el segundo contribuyente más importante

La capacidad, experiencia y cohesión del **equipo de desarrollo** afecta los resultados del proyecto

La capacidad de los analistas y de los programadores

La experiencia en el área de negocios de la aplicación, en el lenguaje, en las herramientas y en la plataforma

La continuidad del personal y la cohesión del equipo

… pueden empeorar las cosas 10 y hasta 20 veces

¿Por qué nos equivocamos tanto al hacer estimaciones?

Información inexacta acerca del proyecto que estamos estimando

Información inexacta acerca de las capacidades de la organización que llevará a cabo el proyecto

Demasiado caos en el proyecto como para permitir estimaciones exactas

Inexactitudes provenientes del mismo proceso de estimación

Fuentes de error al hacer estimaciones

características del proyecto

- El cono de incertidumbre
- Desarrollos caóticos
- Requisitos inestables

características de las prácticas de estimación

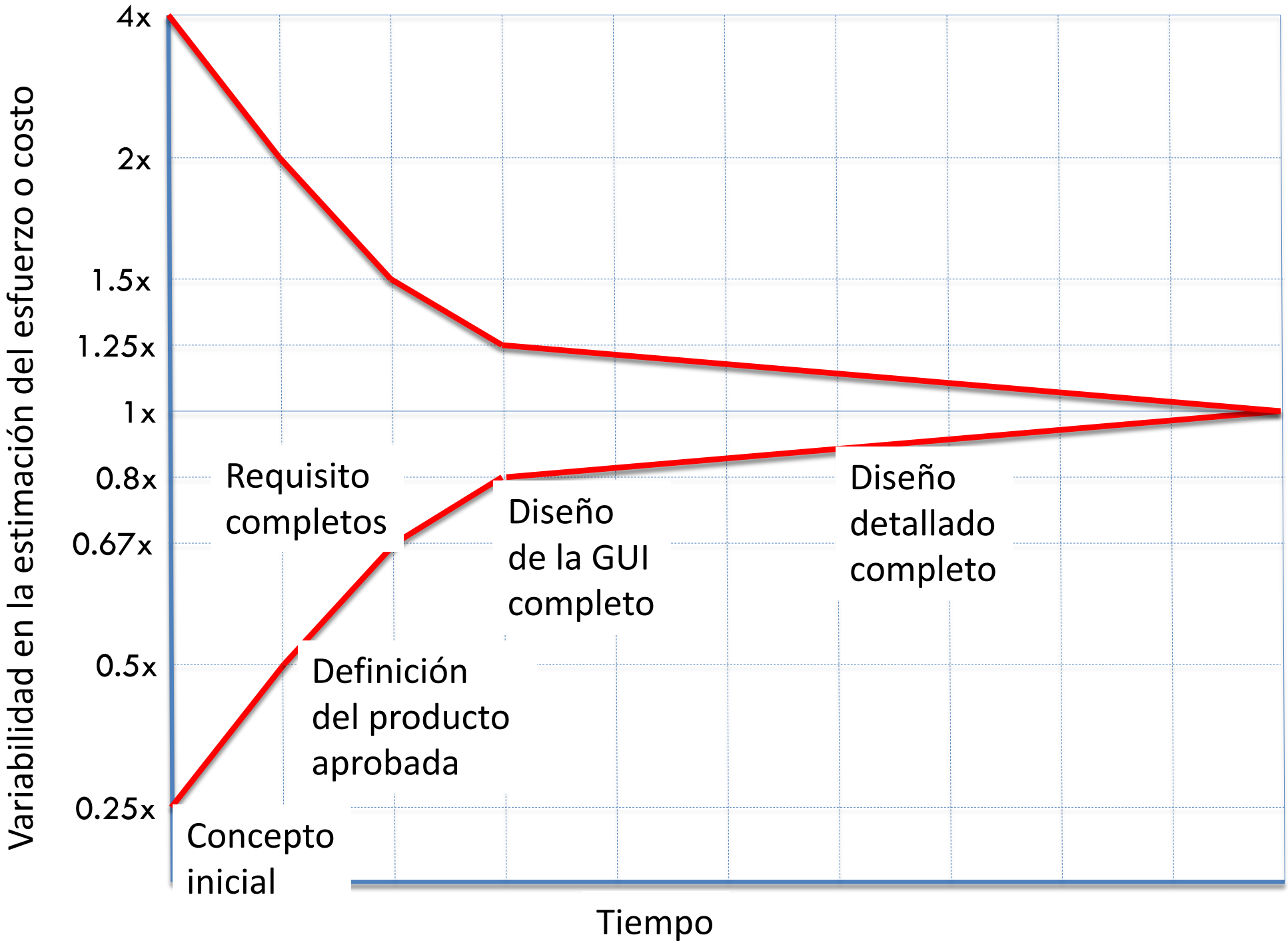
- Actividades omitidas
- Optimismo sin fundamento
- Subjetividad y sesgo
- Estimaciones sacadas “de la manga”
- Precisión injustificada

El cono de incertidumbre: Cómo mejoran las estimaciones a medida que el proyecto avanza

En un proyecto, las estimaciones están sujetas a **cantidades predecibles de incertidumbre** en cada una de sus etapas

Las estimaciones se vuelven más exactas a medida que el proyecto progresa (ver próx. diapositiva):

- las estimaciones hechas al definir el concepto inicial del producto —al inicio del proyecto— tienen una inexactitud de *al menos* un factor de 4
- las estimaciones hechas al completar el diseño de la GUI —cuando el proyecto ha avanzado un 30%— tienen una inexactitud de *al menos* un factor de 1.25


Variabilidad en la estimación del esfuerzo o costo
4x
2x
1.5x
1.25x
1x
0.8x
0.67x
0.5x
0.25x
Requisito completos
Diseño de la GUI completo
Diseño detallado completo
Definición del producto aprobada
Concepto inicial
Tiempo

El cono representa la **exactitud en el mejor caso** que podría tenerse en cada etapa

Estimaciones hechas por estimadores hábiles

Proyectos bien controlados

El cono se estrecha en la práctica sólo si vamos tomando decisiones que eliminen la variabilidad en el proyecto:

- definir la visión del producto, incluyendo comprometerse a qué no se va a hacer
- definir los requisitos, de nuevo, incluyendo lo que no se va a hacer
- diseñar la GUI

Los desarrollos caóticos se ven reflejados en hechos como los siguientes

Los requisitos no fueron bien investigados y los usuarios finales no participaron en su validación

Malos diseños y malas prácticas de programación obligan a muchísima corrección de errores

La planificación del proyecto está incompleta o mal hecha, o bien es abandonada bajo presión

Personal sin experiencia

Integrantes del equipo con personalidad de *prima donna*

Los requisitos inestables presentan dos desafíos de estimación específicos

Son una forma particular de caos en el proyecto, que dificulta o impide que se estreche el cono de incertidumbre

A menudo no se les hace seguimiento ni se reestima el proyecto cuando corresponde

Actividades a veces omitidas al hacer las estimaciones

Requisitos:

- conversiones de datos, sistemas de ayuda, interfaces con sistemas externos

Actividades de desarrollo de software:

- aprendizaje de nuevas herramientas, puesta al día de nuevos integrantes del equipo, instalación del producto, revisión de la documentación técnica

Optimismo sin fundamento

“Vamos a ser más productivos en este proyecto que lo que fuimos en el anterior”

“Muchas cosas salieron mal en el último proyecto; pero muchas menos van a salir mal en éste”

“Empezamos este proyecto lentamente, enfrentando una curva de aprendizaje empinada, pero lo que hemos aprendido nos permitirá terminar el proyecto más rápido”

El **análisis de puntos de función** es un método para medir la magnitud de un sistema

Un **punto de función** es una unidad de medida sintética del tamaño o magnitud de una aplicación de software

Los puntos de función son más fáciles de calcular —a partir de una especificación de requisitos— que, p.ej., las líneas de código

… y proporcionan una base para calcular la magnitud en líneas de código

El método es aplicable a diversos tipos de software

Fue desarrollado en un entorno de sistemas de información:

- sus pautas están descritas en el lenguaje del desarrollador de sistemas de información
- es suficientemente flexible para ser adaptable a otros entornos de software —científico, tiempo real, telecomunicaciones

El *International Function Point Users Group* (IFPUG) promulga el Counting Practices Manual, con los estándares y pautas vigentes para el proceso de conteo

El método nació a mediados de los 70’s en IBM

… debido a la necesidad de establecer una medida eficaz y predecible de la magnitud de un sistema,

… que pudiera usarse para predecir o estimar mejor la entrega de software

Allan Albrecht lo presentó en 1979 en una conferencia

Contar puntos de función toma seis pasos

1) Determinar la frontera de la aplicación

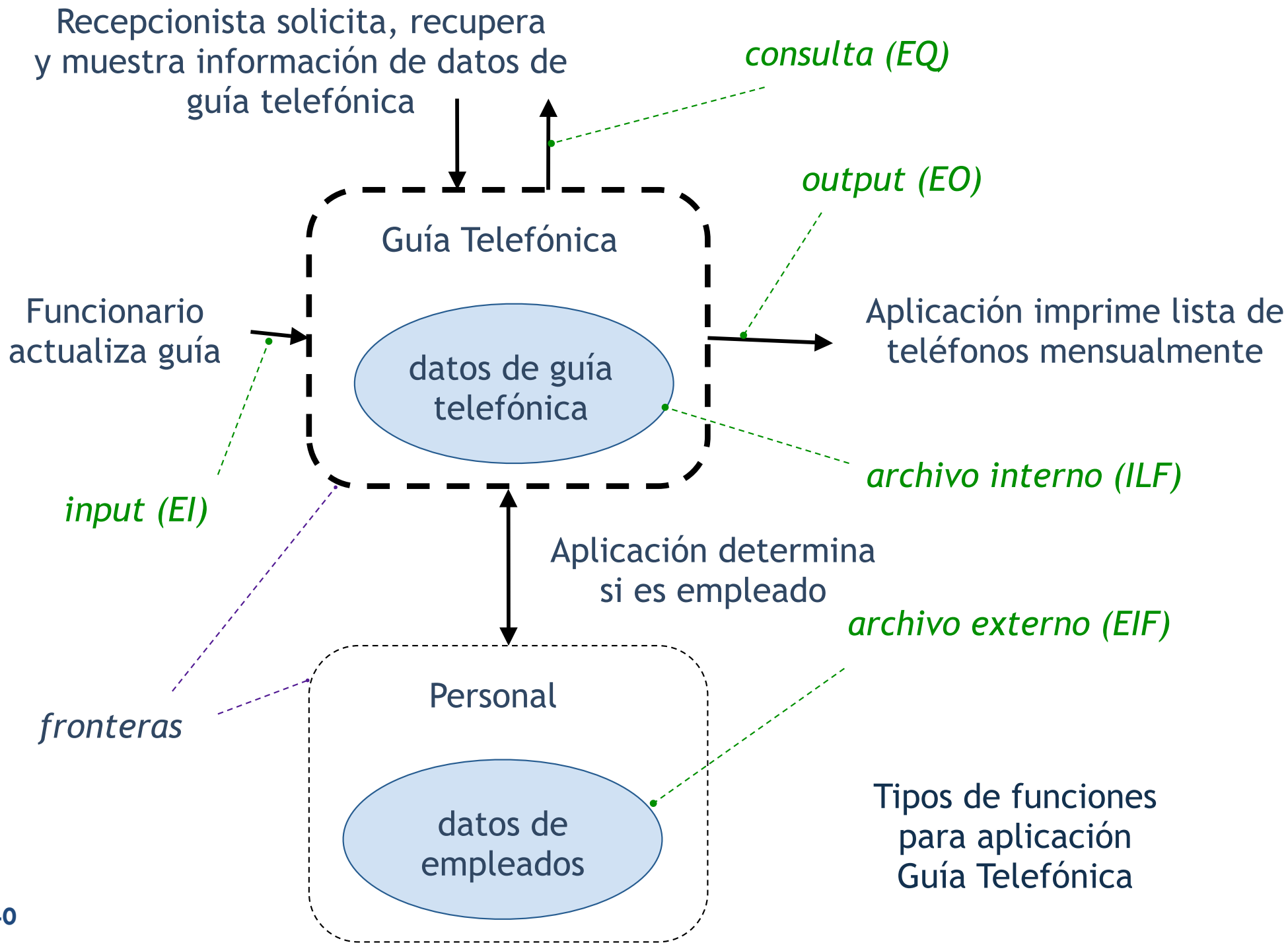
2) Identificar los archivos lógicos internos y los archivos de interfaz externos, y sus niveles de complejidad

3) Identificar los *inputs*, los *outputs* y las *consultas*, y sus niveles de complejidad

4) Calcular los puntos de función no ajustados, sumando los resultados de 2) y 3)

5) Calcular el factor de ajuste, basado en 14 características generales del sistema

6) Calcular los puntos de función ajustados, multiplicando los puntos de función no ajustados, de 4), por el factor de ajuste, de 5)


Recepcionista solicita, recupera y muestra información de datos de guía telefónica
consulta (EQ)
output (EO)
Guía Telefónica
Funcionario actualiza guía
datos de guía telefónica
Aplicación imprime lista de teléfonos mensualmente
archivo interno (ILF)
input (EI)
Aplicación determina si es empleado
archivo externo (EIF)
fronteras
Personal
datos de empleados
Tipos de funciones para aplicación Guía Telefónica

¿Qué representan los puntos de función?

Los puntos de función debieran representar la **funcionalidad que el usuario solicita** —puntaje por funcionalidad:

usuario final, personal de marketing, analista de negocios, comprador

¿Cómo se usan los puntos de función?

La magnitud de una aplicación en puntos de función puede usarse —junto con otros atributos del proyecto— para estimar costo y recursos necesarios:

- nivel de habilidad de los desarrolladores, lenguajes de programación, métodos y técnicas a emplear, tareas a ejecutar

¿Cuándo se cuentan los puntos de función?

Durante el desarrollo, la información disponible para ayudar a contar puntos de función aumenta, incluyendo:

- propuesta del proyecto
- documento de requisitos
- diagramas del sistema y de entidad-relación
- especificaciones funcionales y del sistema
- modelos lógico de datos y de procesos
- especificaciones de programas y módulos
- manual del usuario
- material de capacitación

Primero, determinamos la *frontera de la aplicación*

Indica los límites o separaciones entre dos contextos:

- la aplicación que está siendo medida
- el dominio del usuario o aplicaciones externas

Se basa en el punto de vista del usuario:

- el usuario debe poder definir el alcance de la aplicación

Luego, contamos los *archivos lógicos internos* …

ILF: Cada grupo principal de datos o de información de control relacionada lógicamente e identificable por el usuario,

… que es controlado (generado, usado o mantenido) completamente por la aplicación:

- un ILF debe ser actualizado por al menos uno de los *inputs* de la aplicación
- un ILF típicamente será leido o referenciado por un *output* o una *consulta* u otro *input*
- p.ej., un archivo plano, una tabla en una base de datos relacional

… y los *archivos de interfaz externos*

EIF: Cada archivo (grupo lógico de datos o de información de control) con el cual la aplica-ción interactúa,

… pero que es controlado (mantenido) por otra aplicación:

- un EIF será leído o referenciado por al menos un *input*, un *output* o una *consulta* de la aplicación

En seguida, contamos los *inputs* …

EI: Cada proceso elemental de la aplicación sobre datos o información de control que entra a la aplicación (desde más allá de su frontera):

- pantallas, formularios, cajas de diálogo o señales de control a través de los cuales un usuario final u otra aplicación agrega, elimina o cambia los datos de la aplicación
- los datos procesados deben mantener uno o más ILF's (la información de control no necesariamen-te)

… los *outputs* …

EO: Cada proceso elemental de la aplicación que genera datos o información de control que sale de la aplicación:

- pantallas, reportes, gráficos o señales de control que la aplicación genera para ser usado por un usuario final u otro programa
- normalmente, procesan, combinan o resumen información compleja y son elaboradamente formateados

… y las *consultas*

EQ: Cada proceso elemental de la aplicación compuesto por una combinación de *input-output* en la que el *input* (información de control que define la consulta) produce un *output* simple inmediato (datos obtenidos de uno o más ILF's o EIF's):

- no se hace mantenimiento a ningún ILF durante el proceso
- se refiere a una búsqueda directa de un dato específico, normalmente usando una sola clave y sin formatear el output
- en aplicaciones Web, la diferencia entre consultas y *outputs* es difusa

A continuación, hay que completar esta tabla …

Característica de la aplicación	Complejidad **baja**	Complejidad **promedio**	Complejidad **alta**
EI's	…… × 3 = ___	…… × 4 = ___	…… × 6 = ___
EO's	…… × 4 = ___	…… × 5 = ___	…… × 7 = ___
EQ's	…… × 3 = ___	…… × 4 = ___	…… × 6 = ___
ILF's	…… × 7 = ___	…… × 10 = ___	…… × 15 = ___
EIF's	…… × 5 = ___	…… × 7 = ___	…… × 10 = ___

… y sumar los 15 valores en los ___ para obtener los **puntos de función no ajustados** (UFP's)

La complejidad de cada ILF y EIF depende de dos factores

1) El número de campos o atributos, reconocibles por el usuario, de cada registro del archivo

2) El número de grupos (tipos de datos), reconocibles por el usuario, en que se pueden clasificar los registros contenidos en el archivo

tipos de datos	campos o atributos		
	1 - 19	20 - 50	> 50
1	baja	baja	promedio
1 - 5	baja	promedio	alta
> 5	promedio	alta	alta

La complejidad de cada EI, EO y EQ depende de dos factores

1) El número de campos o atributos que cruzan la frontera de la aplicación

2) El número de archivos referenciados

archivos referenciados EI / EO y EQ	campos o atributos EI / EO y EQ		
	1-4 / 1-5	5-15 / 6-19	> 16 / > 20
0 o 1	baja	baja	promedio
2 / 2 o 3	baja	promedio	alta
> 2 / > 3	promedio	alta	alta

Características generales del sistema que afectan la cuenta

1) Comunicación de datos

2) Procesamiento distribuido de datos (dentro de la frontera)

3) (Requisitos de) Desempeño

4) Configuración de explotación intensamente usada

5) Tasa de transacciones

6) Entrada de datos on-line

7) Eficiencia para el usuario final

8) Actualización en línea

9) Procesamiento complejo

10) Reusabilidad

11) Facilidad de instalación

12) Facilidad de operación

13) Múltiples sitios

14) Facilitación de cambios

Cada una de las 14 características aporta según su *grado de influencia*

No está presente o no influye: 0

La influencia es incidental: 1

La influencia es moderada: 2

La influencia es mediana: 3

La influencia es significativa: 4

La influencia es total: 5

Cálculo de los **puntos de función ajustados**

La suma de los aportes de las 14 características es su **grado total de influencia** (TDI)

El **factor de ajuste del valor** (VAF) se calcula como

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

Finalmente, el número de **puntos de función ajustados** (FP) se calcula como

$$FP = VAF \times UFP$$