

Estimación

Ejecutivo: Cuanto tiempo crees que tomará el proyecto?
Necesitamos el software listo en 3 meses para mostrarlo en la expo. No puedo darte mas gente así que sería con el equipo actual.

Jefe de Proyecto: OK déjame hacer unos cálculos y te respondo

al día siguiente ...

Jefe de Proyecto: Hemos estimado que nos tomará 5 meses

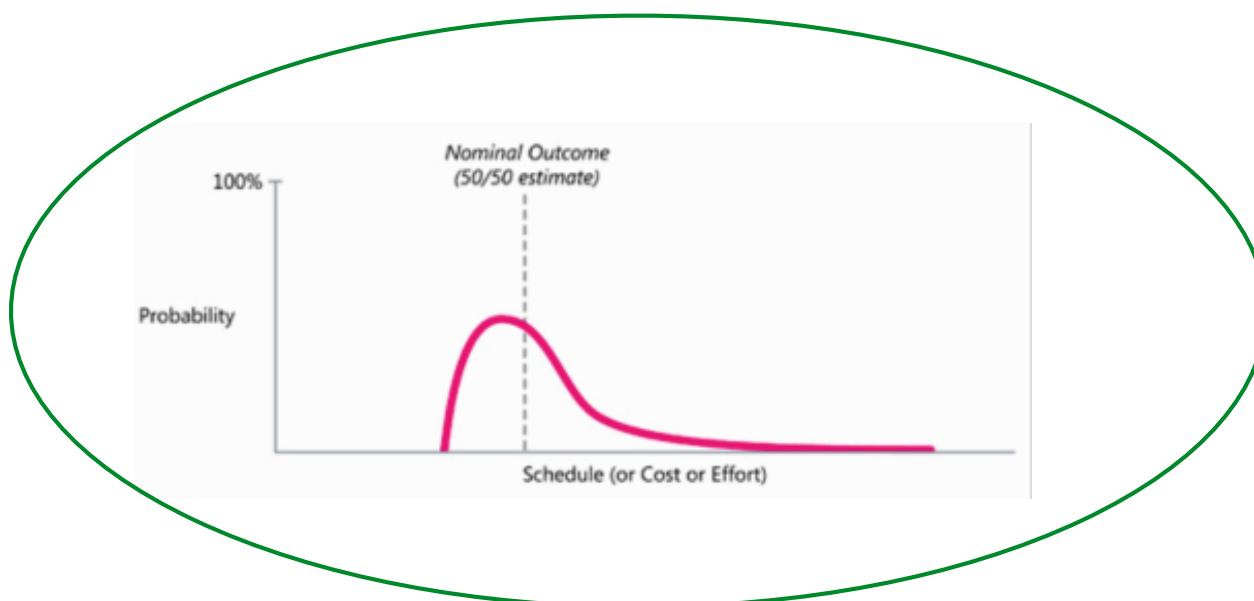
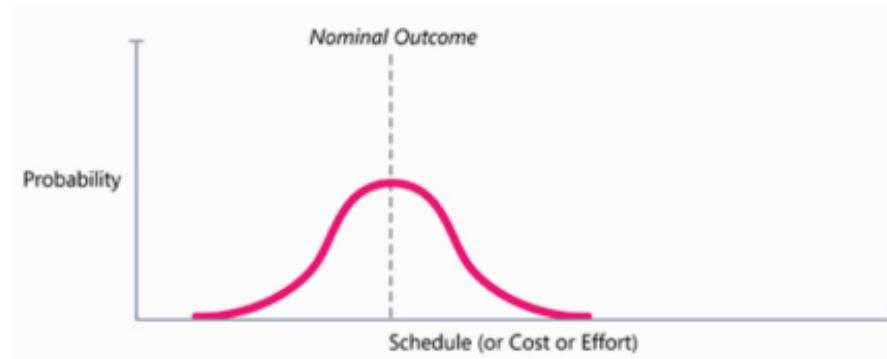
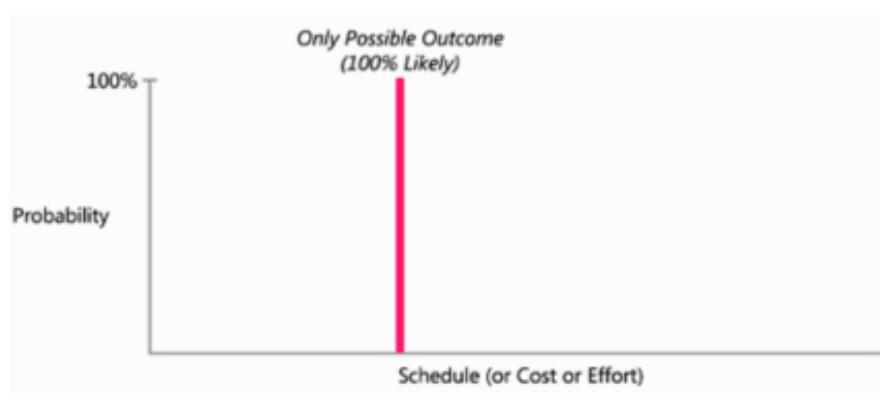
Ejecutivo: 5 meses ? No me pusiste atención ? Te dije que lo necesitabamos en 3 meses

Estimación, Meta y Compromiso

- ▶ En el ejemplo anterior los 3 meses era una **meta**
- ▶ Los 5 meses era la **estimación**
- ▶ Se le pide al jefe de proyecto un **compromiso**

Lo que se le estaba pidiendo al jefe de proyecto no era una estimación sino un plan para lograr la meta

Estimaciones son afirmaciones probabilísticas



Estimación y Control de Proyecto

- ▶ Una vez que se hace la estimación y se acepta el compromiso es necesario controlar el proyecto para cumplir con la meta
- ▶ Muchas cosas cambian durante la ejecución del proyecto
 - ▶ se agregan o cambian requerimientos
 - ▶ parte del equipo de va o se usa en otras tareas
 - ▶ equipo era menos experimentado de lo pensado
 - ▶ etc

El ejemplo de la maleta



- ▶ Te quieres ir de viaje solo con el equipaje de mano
- ▶ Pones tu ropa al lado de la pequeña unidad de "equipaje de mano" y pareciera que "casi" cabe
- ▶ Si falta poco, empacar muy cuidadosamente o dejar algo no indispensable fuera puede funcionar
- ▶ Si sentarse sobre la maletita no hace el truco tendrás que optar
 - ▶ dejar parte de la ropa en casa
 - ▶ usar la maleta mas grande

La lección

- ▶ Cuando la diferencia entre la meta y la estimación es pequeña (< 20%) puede asumirse la meta como compromiso
- ▶ Ello va a requerir un control y una planeación muy cuidadosa
- ▶ El propósito de la estimación es saber si un trabajo cuidadoso hará posible o no cumplir con la meta
- ▶ Estimación inicial no corresponde al proyecto que terminará ocurriendo

Una buena estimación

Es aquella que provee una vista suficientemente clara de la realidad del proyecto como para permitir que el líder del proyecto pueda tomar buenas decisiones sobre como controlar el proyecto para lograr la meta

Sobreestimar vs Subestimar

Problemas de sobreestimar

- ▶ ley de Parkinson - el trabajo se expandirá hasta usar todo el tiempo disponible
- ▶ ley de Goldratt (síndrome del estudiante) - si hay demasiado tiempo se malgasta al comienzo del proyecto

Sobreestimar vs Subestimar

- ▶ Problemas de Subestimar
 - ▶ errores de planeación, errores de coordinación
 - ▶ reducción de tiempos en análisis y diseño puede generar mucho mas atraso
 - ▶ dinámica de proyecto atrasado: reuniones para reagendar, para recuperar tiempo, etc
 - ▶ problemas con el cliente
 - ▶ arreglos parches (no hay tiempo)

Lo que deben recordar ...

La penalización por sobreestimar es mucho menos que la penalización por subestimar



El gran problema

La industria del software no tiene un problema de estimaciones sino que un problema de subestimación

Subestimación



Las 4 fuentes de error en estimaciones

- ▶ Información inprecisa sobre el proyecto
- ▶ Información inprecisa sobre las capacidades del equipo de desarrollo
- ▶ Proyecto demasiado caótico (blanco móvil)
- ▶ Imprecisiones del proceso de estimación mismo

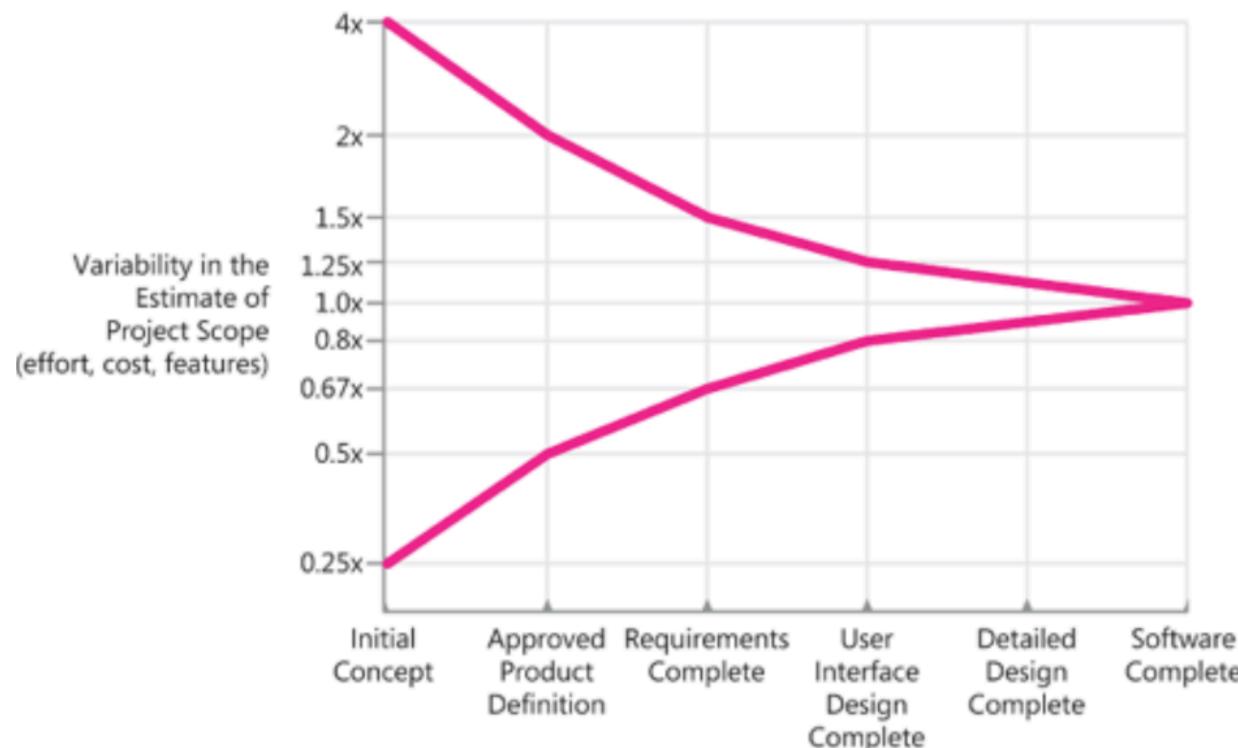
Información del Proyecto

- ▶ Al comienzo del proyecto no se tiene claridad respecto a lo que se debe construir
- ▶ Cada pieza de software tiene muchas formas de ser implementada (performance, usabilidad, etc)
- ▶ Esto produce una enorme gama de posibilidades

El cono de incertidumbre

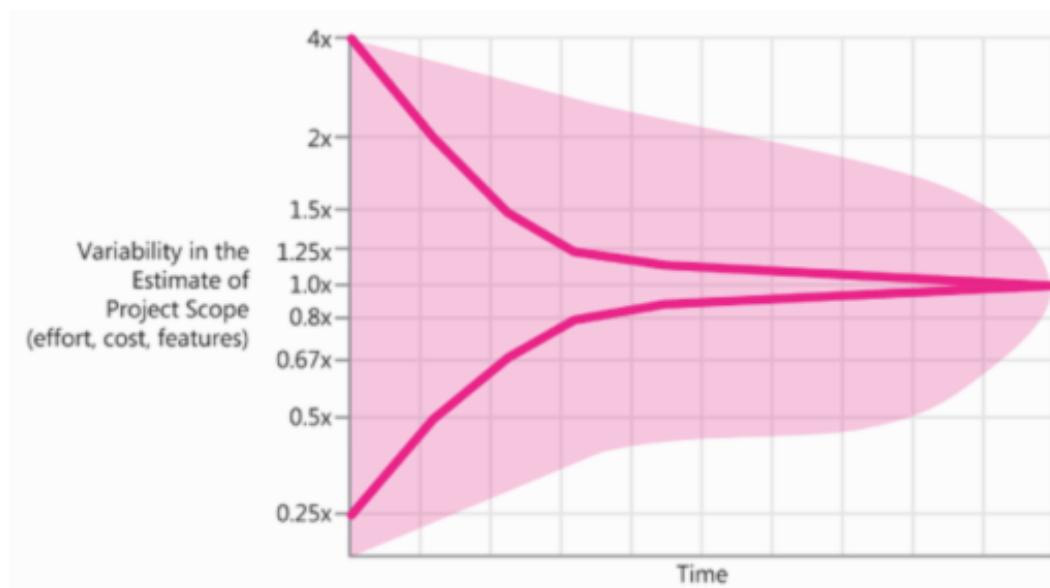
A medida que el proyecto avanza se va reduciendo el número de posibilidades y aumentando la certeza en la estimación

Cono representa el "best case" en términos de precisión



El cono no se reduce solo ...

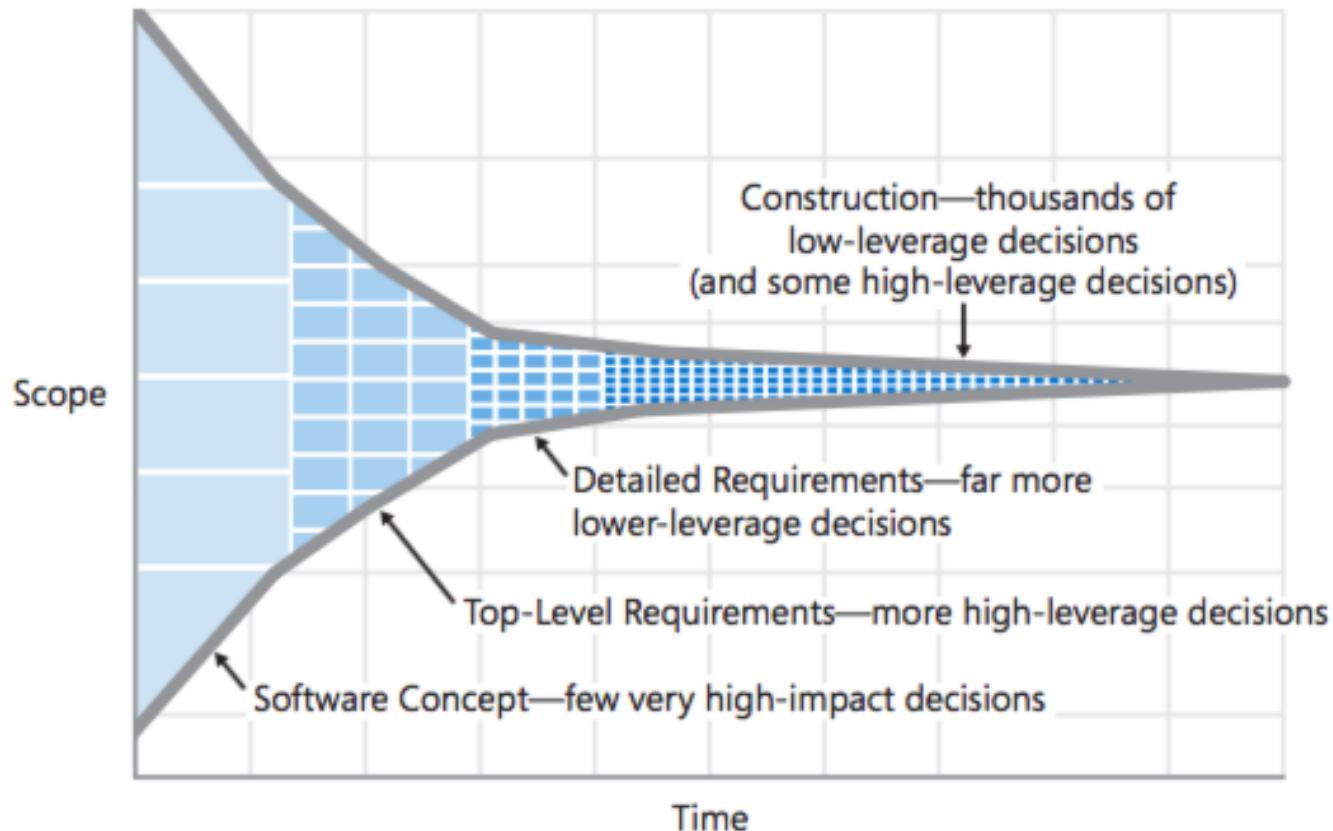
- ▶ Es necesario hacer un esfuerzo para ir reduciendo la incertidumbre
- ▶ Si ello no ocurre puede ocurrir una nube de incertidumbre



Descomposición ayuda mucho

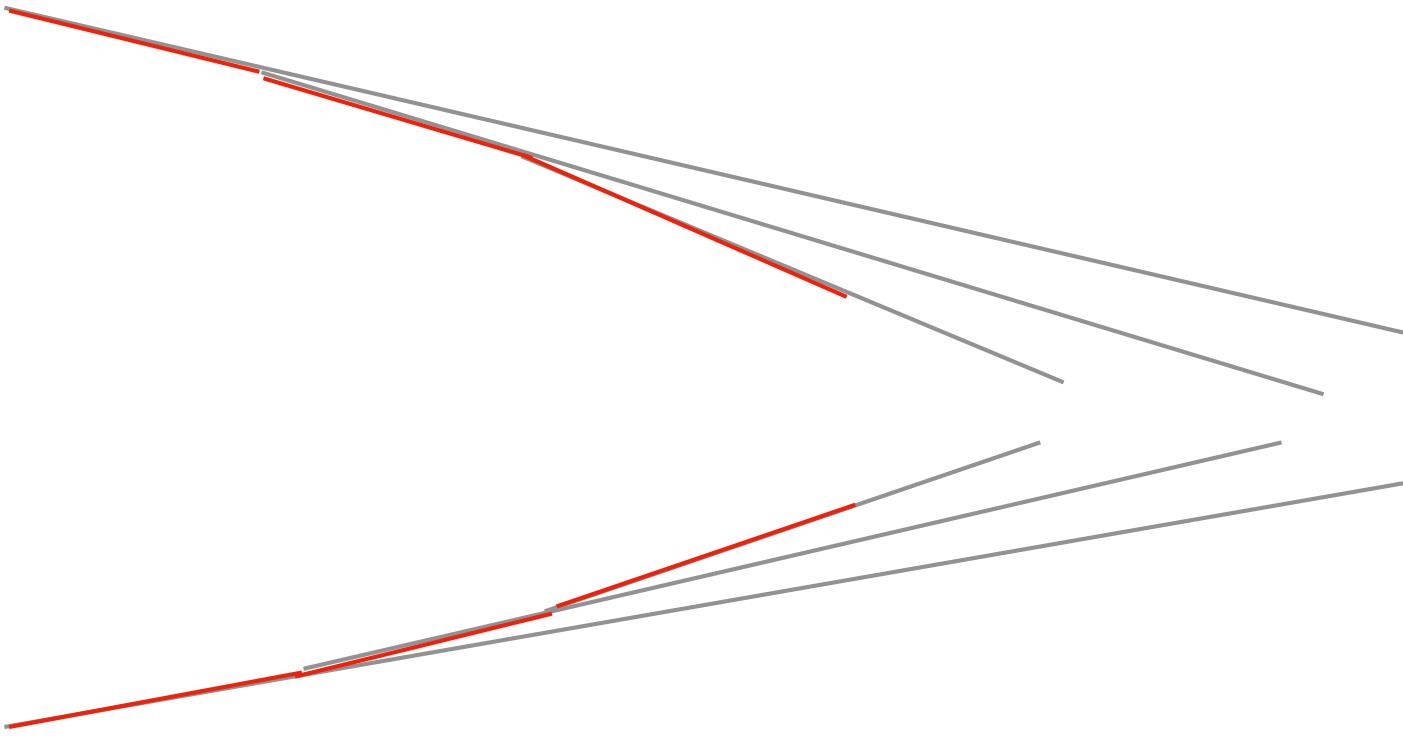
- ▶ Beneficios de ley de los grandes números
 - ▶ sobrestimación en unas se compensa con subestimación en otras
- ▶ Al principio son unidades mayores y se achican a medida que se avanza en el cono

Cono y Descomposición



Desarrollo Iterativo

- ▶ Hay una secuencia de conitos pero cada vez parte con un diámetro menor
- ▶ Una manera de reducir rápidamente es trabajar más en análisis y diseño en primera iteración aún cuando solo se implemente una parte de ello
- ▶ Por ejemplo la mayor parte de los relatos de usuario se tienen al comienzo



Recomendaciones Generales

- ▶ No comprometerse con estimaciones improvisadas
- ▶ No usar una precisión que genere demasiadas expectativas

La dañina respuesta improvisada

Ejecutivo: ¿Cuanto tiempo crees que tomaría implementar un preview?

Tu: No lo sé. Me magino que una semana

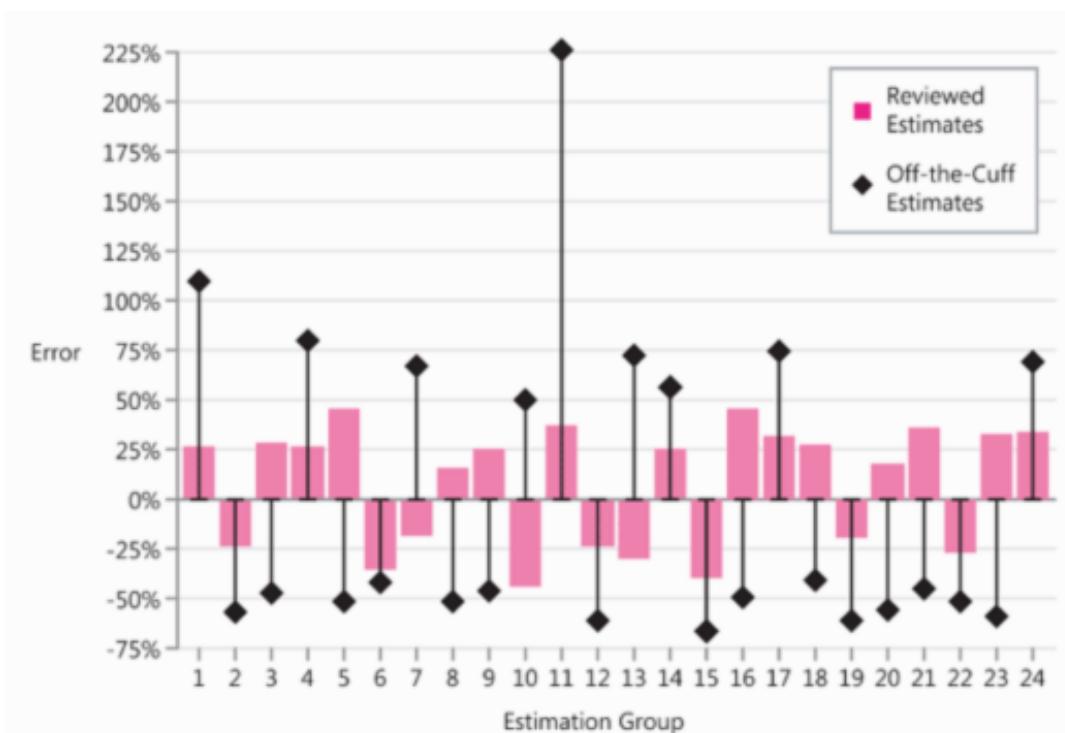
Después de verlo con cuidado te das cuenta que no consideraste algunos problemas y eran 3 semanas ...

Vas donde el ejecutivo pero antes de poder abrir la boca ...

Ejecutivo: Ya te conseguí aprobación en la reunión de presupuesto de esta tarde, todos están ansiosos de ver esta nueva feature la próxima semana. ¿Podrías comenzar hoy mismo?

La respuesta improvisada suele ser pésima

Es mejor esperar aunque sean 20 minutos y hacer un pequeño análisis y NO DAR estimaciones "al tufo"



Precisión

- ▶ Si indicas que el proyecto tomará 90.5 días los clientes asumen que estimación podría variar tal vez en medio día
- ▶ Mejor indicar 3 meses o 1 trimestre
- ▶ Precisión - número de dígitos significativos
- ▶ Exactitud - cuan cerca del valor real
- ▶ Ejemplo: valor de pi
 - ▶ 3
 - ▶ 4.14159

¿ Que es lo que queremos estimar ?

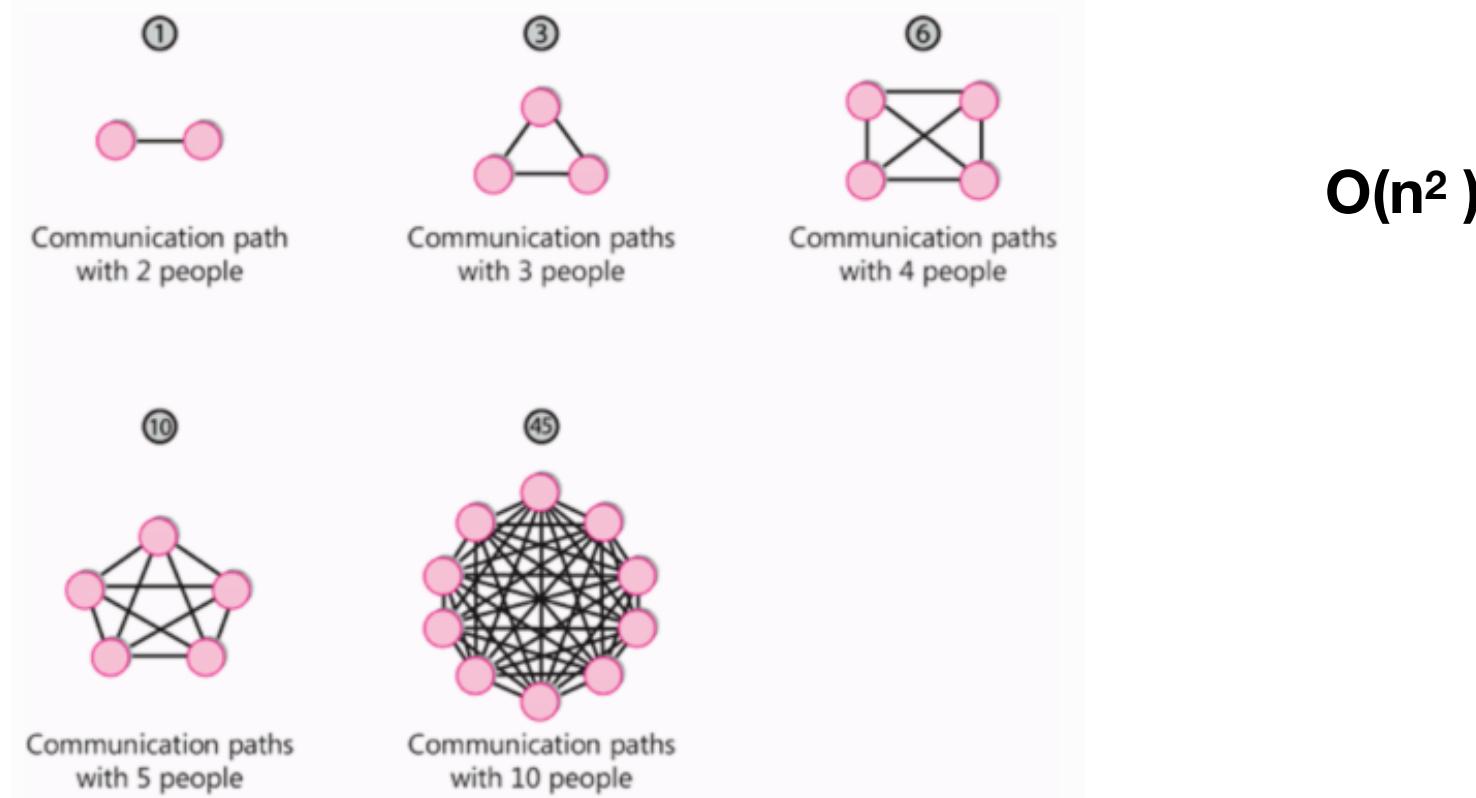
- ▶ Esfuerzo de desarrollo - cuanta gente durante cuanto tiempo (staff months)
- ▶ Tiempo de desarrollo (weeks, months)

Se parte con estimación de tamaño

- ▶ El tamaño es el factor mas relevante en determinar el tiempo y el esfuerzo de un proyecto
- ▶ Considerar que aquí rige la **deseconomía** de escala
 - ▶ mientras mas grande menor productividad
 - ▶ un producto de 100.000 LOC tomará más de 10 veces lo que toma hacer uno de 10.000 LOC

Origen de la deseconomía de escala

- ▶ Necesidad de comunicación y sincronización



¿ Y que demonios es una loc ?

- ▶ loc - line of code
- ▶ número de líneas de código fuente
- ▶ la métrica mas utilizada para tamaño
- ▶ simple
- ▶ gran ventaja es que una vez construido se puede medir en forma exacta
- ▶ puede incluirse algunos ajustes como
 - ▶ no contar las líneas de comentario
 - ▶ no contar las líneas en blanco

Problema de las locs

- ▶ Dependencia de
 - ▶ lenguaje de programación
 - ▶ estilo de escritura del código
 - ▶ calidad del código (mejor tiende a ser más corto)
 - ▶ comentarios y líneas en blanco
- ▶ Ventaja: una vez construido se puede medir en forma exacta

Hello world

```
IDENTIFICATION DIVISION.  
PROGRAM-ID. HELLO-WORLD.  
* simple hello world program  
PROCEDURE DIVISION.  
    DISPLAY 'Hello world!'  
    STOP RUN.
```

COBOL

```
public class HelloWorld {  
  
    public static void main(String[] args) {  
        // Prints "Hello, World" to the terminal window.  
        System.out.println("Hello, World");  
    }  
}
```

JAVA

```
print('Hello, world!')
```

Python

```
puts "Hello World!"
```

Ruby

```
console.log('Hello world!')
```

JavaScript

```
#include <stdio.h>  
int main()  
{  
    // printf() displays the string inside quotation  
    printf("Hello, World!");  
    return 0;  
}
```

C

Otras métricas de tamaño que pueden usarse en forma temprana

- ▶ pueden usarse en forma temprana
- ▶ no dependen del lenguaje ni del estilo de programación
- ▶ puntos de función
- ▶ user stories
- ▶ user story points

Puntos de Función

Independiente de lenguaje o estilo de programación

Tipo	Baja	Mediana	Alta
Entradas	x3	x4	x6
Salidas	x4	x5	x7
Consultas	x3	x4	x6
Archivos Internos (tablas)	x7	x10	x15
Archivos Externos (tablas)	x5	x7	x10

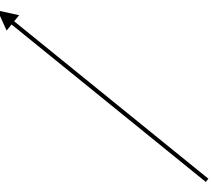
Puntos de Función no Ajustados: PFNA

Factor de Complejidad (14 factores) : 0.65 - 1.35

Puntos de Función Ajustados = PFNA * FC

Factor de Complejidad

- 01 comunicaciones
 - 02 funciones distribuidas
 - 03 objetivos de desempeño
 - 04 configuración sobrecargada
 - 05 tasa de transacciones
 - 06 entrada de datos on line
 - 07 eficiencia para usuario
 - 08 actualización en línea
 - 09 proceso complejo
 - 10 reuso
 - 11 facilidad de instalación
 - 12 facilidad de operación
 - 14 varios sitios
 - 14 acilidad de mantención
- Mín 14 * 0 = 0
Máx 14 * 5 = 70
- $$FC = 0.65 + N/100$$



Líneas de Código vs Puntos de Función

Lenguaje	Nivel	LOC/FP
Ada 95	6.50	49
ANSI SQL	25.00	13
APL*PLUS	10.00	32
Assembly (Basic)	1.00	320
BASIC A	2.50	128
C	2.50	128
C++	6.00	53
CLIPPER DB	8.00	40
COBOL	3.00	107
DELPHI	11.00	29
EIFFEL	15.00	21
FORTRAN 90	4.00	80
Haskell	8.50	38
JAVA	6.00	53
LISP	5.00	64
MODULA 2	4.00	80
Objective-C	12.00	27
PASCAL	3.50	91
PERL	15.00	21
PowerBuilder	20.00	16
PROLOG	5.00	64
RPG III	5.75	56
SCHEME	6.00	53
SMALLTALK 80	15.00	21
TCL	5.00	64
Visual Basic 5	11.00	29
Visual C++	9.50	34

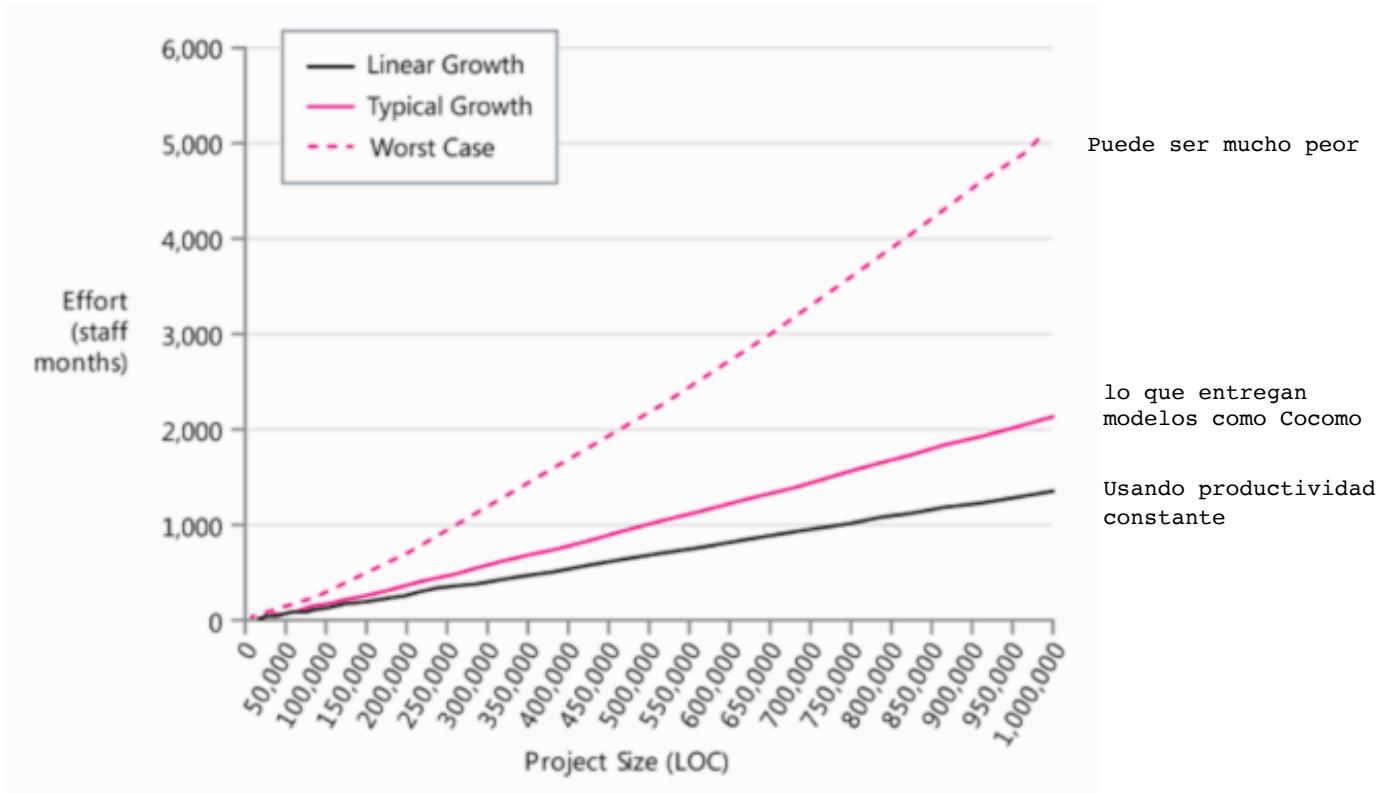
User stories y storypoints

- ▶ Número de relatos de usuario es una buena indicación del tamaño (complejidad) del software
- ▶ Si los relatos varían mucho en complejidad es mejor incluir ponderadores lo que da origen a los storypoints
- ▶ Se asignan ponderadores en escalas no lineales porque esfuerzo aumenta en forma no lineal
 - ▶ potencias de 2 : 1, 2, 4, 8, 16
 - ▶ secuencia de fibonacci: 1, 2, 3, 5, 8

Del tamaño al esfuerzo

- ▶ Métrica de esfuerzo es el "hombre-mes" (staff months)
- ▶ Productividad individual varía mucho (un orden de magnitud)
- ▶ Usar cifras de productividad histórica
 - ▶ Ej: este tipo de proyecto - 620 loc por hombre-mes
 - ▶ Sin embargo, recordar que hombres y meses no son intercambiables ("The Mythical Man-Month" by Fred Brooks)

Evidencia



Productividades Típicas

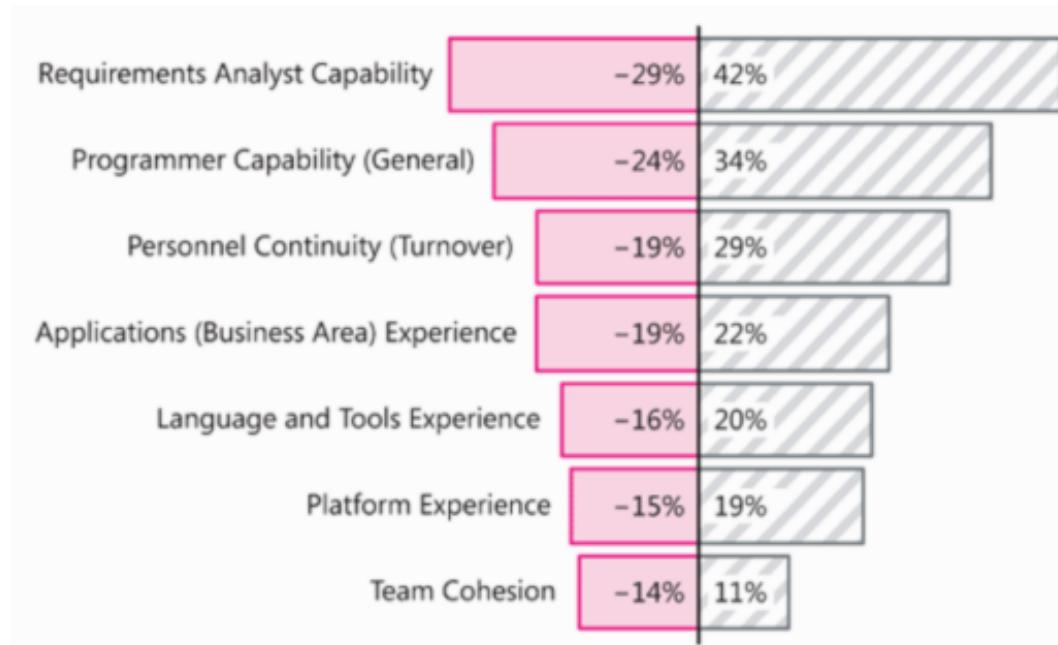
Project Size (in Lines of Code)	Lines of Code per Staff Year (Cocomo II Nominal in Parentheses)
10K	2,000–25,000 (3,200)
100K	1,000–20,000 (2,600)
1M	700–10,000 (2,000)
10M	300–5,000 (1,600)

Los modelos dan cuenta de las variaciones en el tipo de proyecto

Kind of Software	LOC/Staff Month Low-High (Nominal)		
	10,000- LOC Project	100,000- LOC Project	250,000- LOC Project
Avionics	100–1,000 (200)	20–300 (50) (600)	20–200 (40) (500)
Business Systems	800–18,000 (3,000)	200–7,000 (600)	100–5,000 (500)
Command and Control	200–3,000 (500)	50–600 (100)	40–500 (80)
Embedded Systems	100–2,000 (300)	30–500 (70) (300)	20–400 (60)
Internet Systems (public)	600–10,000 (1,500)	100–2,000 (300)	100–1,500 (200)

Efecto del Equipo Humano

Un proyecto puede requerir hasta 22 veces mas tiempo dependiendo de la calidad de los desarrolladores



La buena noticia ...

- ▶ Muchas veces un grupo desarrolla proyectos de tamaño similar (hasta 2 o 3 veces)
- ▶ Equipo no cambia demasiado
- ▶ Si se mantienen ambas cosas, lo mas probable es que la productividad se mantenga y puede ser usada para la estimación

Efecto del Lenguaje de Desarrollo

- ▶ Menos relevante pero considerar que puede afectar en varias formas
 - ▶ experiencia del equipo con el lenguaje y herramientas asociadas
 - ▶ mayor o menos funcionalidad por línea de código
 - ▶ riqueza de herramientas, librerías y frameworks
 - ▶ trabajo con lenguaje interpretado es más rápido que con uno compilado

Tips generales

- ▶ Siempre tratar de contar en lugar de solo estimarlo
- ▶ Si no es posible contar lo que queremos, ver si podemos contar otra cosa y computar lo que queremos mediante calibración
- ▶ Solo como último recurso usar el juicio

La crucial información histórica

- ▶ Permite hacer la calibración de algo que se puede contar en el valor de lo que queremos estimar
- ▶ Lo mejor es usar información histórica del mismo proyecto
- ▶ Lo segundo mejor es información histórica de la organización
- ▶ Finalmente usar información de la industria (ojalá del país)

Info histórica necesaria	
Features	<ul style="list-style-type: none">▪ Average effort hours per feature for development and/or testing
Stories	<ul style="list-style-type: none">▪ Average total effort hours per story▪ Average number of stories that can be delivered in a particular amount of calendar time
Function Points	<ul style="list-style-type: none">▪ Average development/test/documentation effort per Function Point▪ Average lines of code in the target language per Function Point
Web pages	<ul style="list-style-type: none">▪ Average effort per Web page for user interface work▪ Average whole-project effort per Web page (less reliable, but can be an interesting data point)

Por qué es importante

- ▶ Permite contrarrestar tendencia natural al optimismo en un proyecto nuevo
 - ▶ esta vez no se nos irá el jefe del proyecto
 - ▶ esta vez tenemos mejor tecnología
 - ▶ esta vez no perderemos tiempo al comienzo
- ▶ Estimación basada en historia nos dice que ocurrirá mas o menos lo mismo

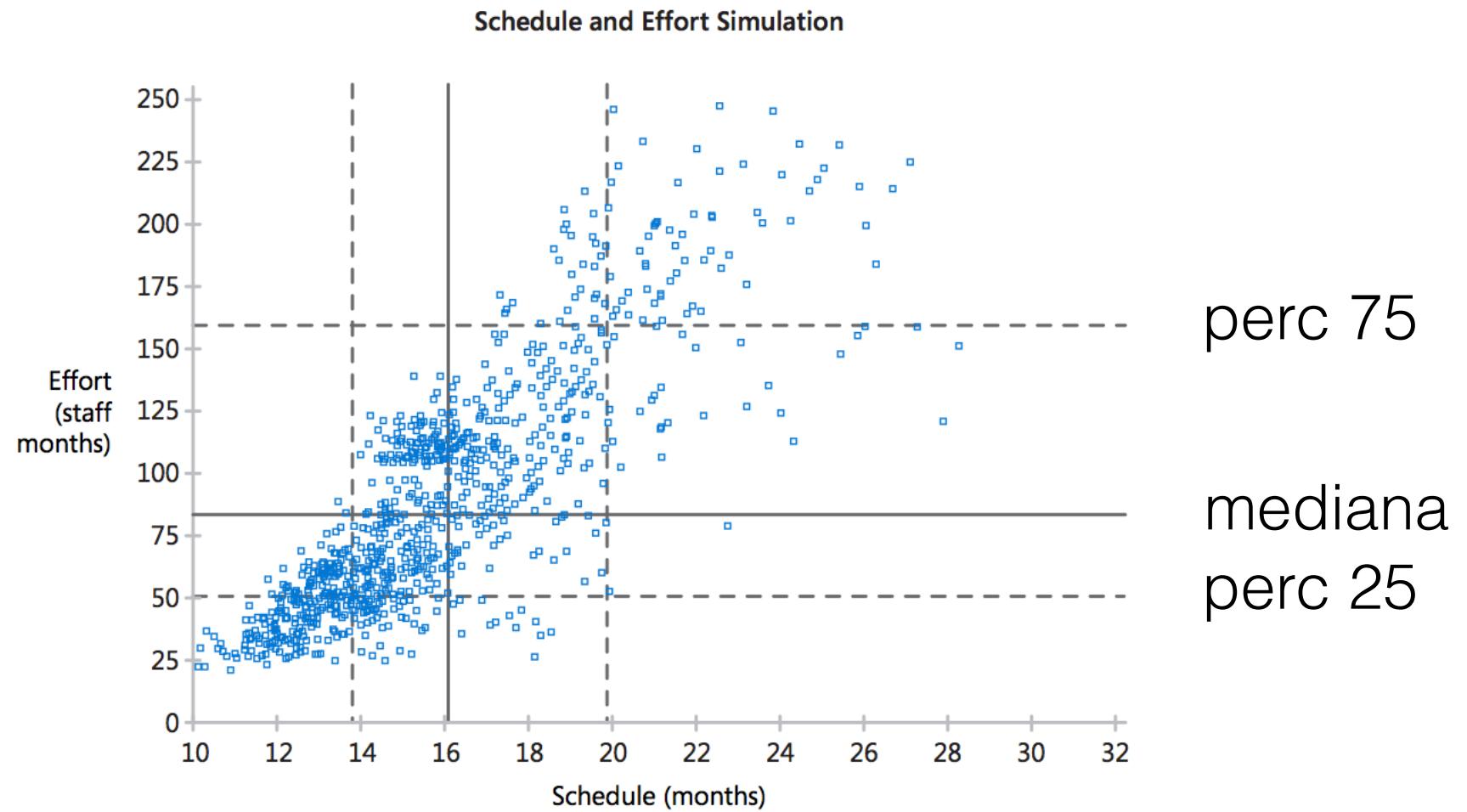
Qué datos recolectar

- ▶ Comenzar con lo más simple
 - ▶ líneas de código una vez completado el proyecto
 - ▶ esfuerzo en personas y meses
 - ▶ tiempo en meses
 - ▶ defectos (calibrados por severidad)
- ▶ Debe recolectarse durante ejecución o inmediatamente terminado (después es difícil)

Calibración: desde cuentas a estimaciones

- ▶ Ya sea que queramos llevar líneas de código, relatos de usuario o casos de uso a esfuerzo o duración se requiere una calibración
- ▶ Fuente para Calibración
 - ▶ datos de la industria
 - ▶ datos históricos de la empresa o del equipo
 - ▶ datos ya generados por el proyecto (muy preciso pero aplicabilidad limitada a iteraciones)

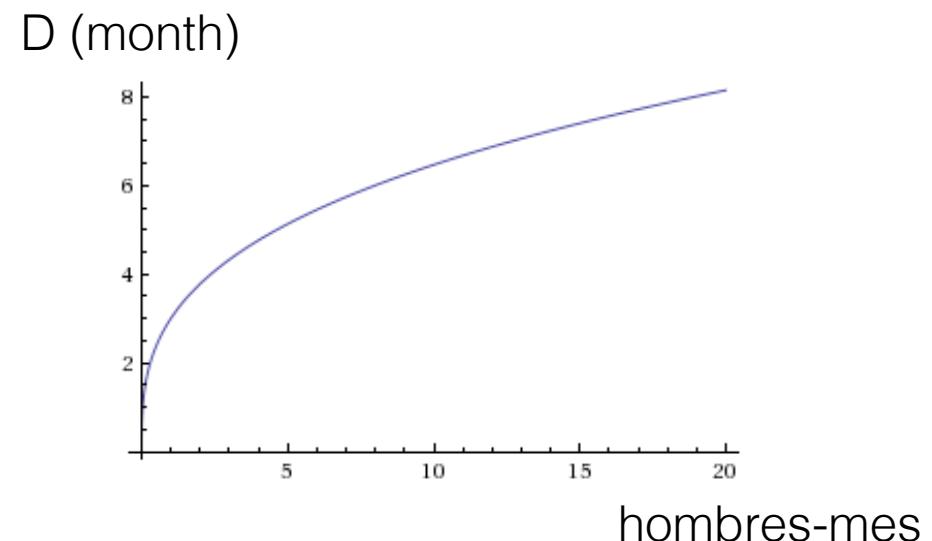
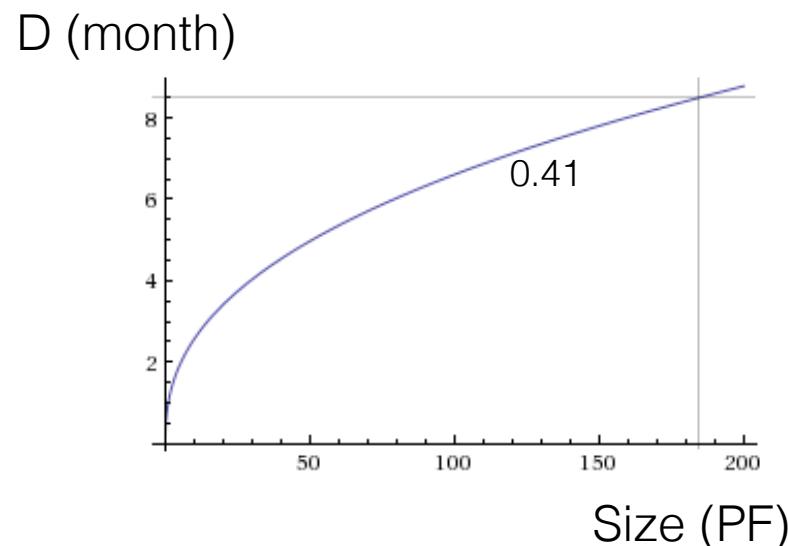
Uso de Datos de la Industria



Modelos basados en Datos de la Industria

$$D = \text{Size}^a \quad a: 0.41 - 0.50 \text{ dependiendo de tipo de sistema}$$

$$D = 3 * E^{1/3}$$



Ejemplo

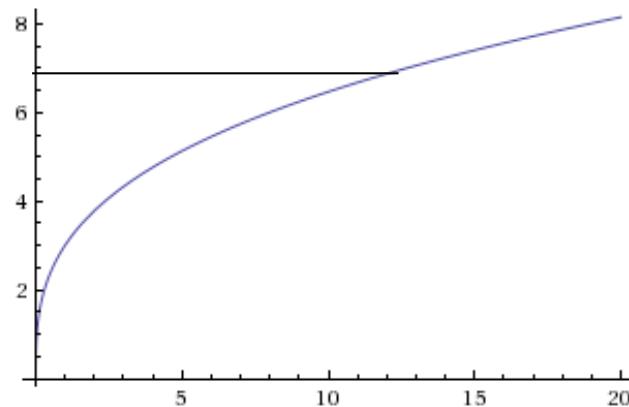
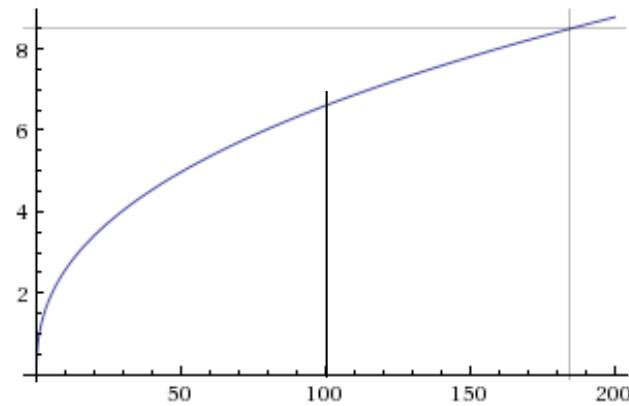
100 puntos de función

Exponente calibrado: 0.41

$$D = 100^{0.41} = 6.6 \text{ meses}$$

$$6.6 = 3 * E^{1/3} \Rightarrow E = 10.6 \text{ hombres mes}$$

Se requieren 2 personas durante 6 meses



Calibración con Info del Proyecto

Story	Points
Story 1	2
Story 2	1
Story 3	4
Story 4	8
...	
Story 60	2
TOTAL	180

Data for Iteration 1

27 story points delivered

12 staff weeks expended

3 calendar weeks expended

Preliminary Calibration

Effort = 27 story points ÷ 12 staff weeks = 2.25 story points/staff week

Schedule = 27 story points ÷ 3 calendar weeks = 9 story points/calender week

Data for Iteration 1

Assumptions (from Preliminary Calibration)

Effort = 2.25 story points/staff week

Schedule = 9 story points/calender week

Project size = 180 story points

Preliminary Whole-Project Estimate

Effort = 180 story points ÷ 2.25 story points/staff week = 80 staff weeks

Schedule = 180 story points ÷ 9 story points/calender week = 20 calendar weeks

¿ Calibración Lineal ?

- ▶ Lo más simple, muchas veces suficiente
 - ▶ equipo de 3 personas puede entregar X relatos implementadas por mes calendario
 - ▶ testers pueden crear un test en X horas
- ▶ Se puede manejar no linealidad con tablas del tipo :

Team Size	Average Stories Delivered per Calendar Month
1	5
2–3	12
4–5	22
6–7	31
8	No data for projects of this size

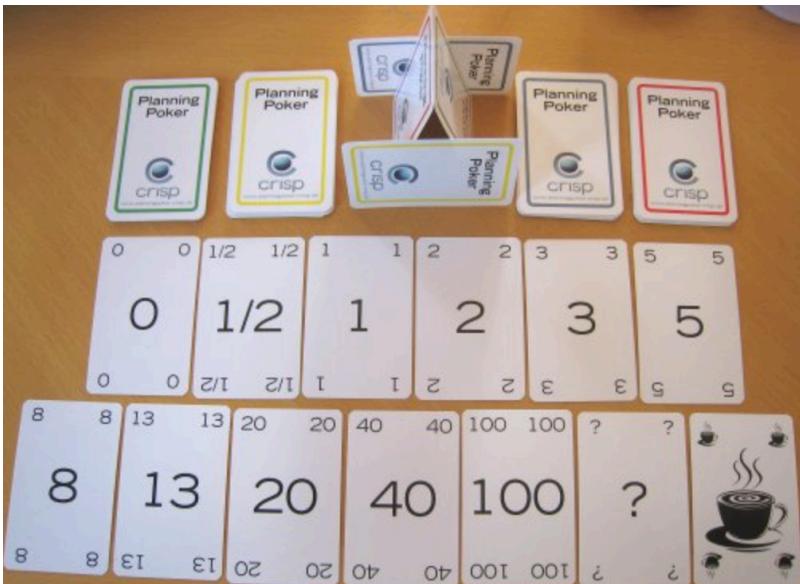
¿Y si le preguntamos a un experto ?

- ▶ Usar juicio de experto como último recurso
- ▶ Por mucha experiencia del experto, una estimación en el vacío será inprecisa
- ▶ Estimación de expertos mejora significativamente cuando hay datos disponibles
- ▶ Apropiada para tareas pequeñas

¿ Y a varios expertos ?

- ▶ Planning poker o Scrum poker
- ▶ Variante del método delphi
- ▶ Para cada relato varios participantes "apuestan" después de analizar el trabajo
- ▶ Las apuestas son con cartas hacia abajo
- ▶ Se levantan las cartas simultáneamente
- ▶ Si hay mucha dispersión se conversa y justifica y se repite hasta llegar a un consenso

Mas detalles



- ▶ Escala de cartas puede variar (Fibonacci, Fibonacci + 1/2, etc)
- ▶ usar un timer para limitar la discusión

Ejemplo de Estimación con Subtareas

Feature	Estimated Days to Complete
Feature 1	1.5
Feature 2	1.5
Feature 3	2.0
Feature 4	0.5
Feature 5	0.5
Feature 6	0.25
Feature 7	2.0
Feature 8	1.0
Feature 9	0.75
Feature 10	1.25
TOTAL	11.25

Estimación de 11.25 días

¿ Exceso de Optimismo ?

Feature	Estimated Days to Complete	
	Best Case	Worst Case
Feature 1	1.25	2.0
Feature 2	1.5	2.5
Feature 3	2.0	3.0
Feature 4	0.75	2.0
Feature 5	0.5	1.25
Feature 6	0.25	0.5
Feature 7	1.5	2.5
Feature 8	1.0	1.5
Feature 9	0.5	1.0
Feature 10	1.25	2.0
TOTAL	10.5¹	18.25

Estimación de 11.25 días está
mucho mas cerca del "best case"

Estimación más real: unos 14 días

Estimación por Analogía

- ▶ Etapa 1: Obtener cifras de tamaño, esfuerzo para proyecto similar
- ▶ Etapa 2: Poner las cifras del nuevo proyecto en términos relativos al anterior
- ▶ Etapa 3: Construir la estimación de tamaño para el proyecto nuevo
- ▶ Etapa 4: Construir estimación de esfuerzo basada en tamaño en comparación al antiguo
- ▶ Etapa 5: Asegurarse que son en verdad comparables / envergadura, tecnología, equipo, etc)

Ejemplo

Aplicación (Web) a Desarrollar: Triad 1.0

Base de Datos - 14 tablas
Interfaz - 19 páginas Web
Gráficos y Reportes - 14 + 16
Clases de Dominio - 15

Aplicación Previa: Accelerator 1.0

Base de Datos - 10 tablas	→	5.000 LOC
Interfaz - 14 páginas Web	→	14.000 LOC
Gráficos y Reportes - 10 + 8	→	9.000 LOC
Clases de Dominio - 15	→	4.500 LOC
Lógica de Negocio - ?	→	11.000 LOC

Subsystem	Actual Size of AccSellerator 1.0	Estimated Size of Triad 1.0	Multiplication Factor
Database	10 tables	14 tables	1.4
User interface	14 Web pages	19 Web pages	1.4
Graphs and reports	10 graphs + 8 reports	14 graphs + 16 reports	1.7
Foundation classes	15 classes	15 classes	1.0
Business rules	???	???	1.5

Subsystem	Code Size of AccSellerator 1.0	Multiplication Factor	Estimated Code Size of Triad 1.0
Database	5,000	1.4	7,000
User interface	14,000	1.4	19,600
Graphs and reports	9,000	1.7	15,300
Foundation classes	4,500	1.0	4,500
Business rules	11,000	1.5	16,500
TOTAL	43,500	-	62,900

Term	Value
Size of Triad 1.0	62,900 LOC
Size of AccSellerator 1.0	÷ 43,500 LOC
Size ratio	= 1.45
Effort for AccSellerator 1.0	× 30 staff months
Estimated effort for Triad 1.0	= 44 staff months