Introducción al PSP (Personal Software Process)

Watts S. Humphrey, 1997

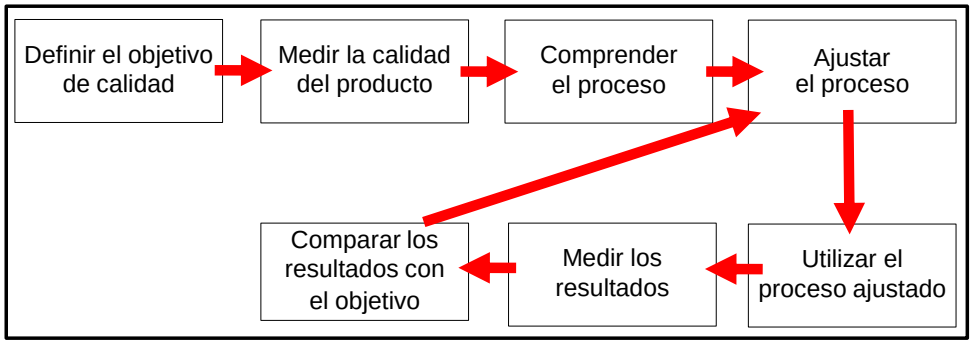
1. El trabajo del ingeniero de Software

- 1.1 ¿Qué es la ingeniería del Software?
  - Planificar el trabajo.
  - Hacer el trabajo de acuerdo con el plan.
  - Esforzarse en productos de máxima calidad.
- 1.2 ¿Por qué es importante una buena ingeniería?
  - Para satisfacer el compromiso costo / planificación, lo que beneficia directamente a la calidad del producto.
- 1.3 El Proceso de Software Personal (PSP)
  - Ayuda a las personas a realizar un buen trabajo
  - Enseña cómo definir, estimar y planear procesos que guiarán el trabajo.

1. El trabajo del ingeniero de Software

- 1. 4 La disciplina del trabajo de alta calidad
  - La disciplina PSP proporciona un marco de trabajo estructurado para desarrollar las habilidades personales y los métodos que necesitará como Ingeniero de Software.
  - La cuestión no es si necesita habilidades personales, sino cuánto tiempo necesita para desarrollarlas y cómo las utilizará de forma consistente.
  - La disciplina PSP acelerará el aprendizaje.
- 1.5 La importancia del trabajo de alta calidad.
  - Para producir software de calidad, cada IS debe trabajar con calidad.
- 1.6 Mejorando la calidad del trabajo
  - Medir, usar la medida para analizar objetivos y, si es necesario, cambiar.

1.7 EL PROCESO DE MEJORA



2. La administración del tiempo

- 2.1 La lógica del manejo del tiempo
  - Probablemente hará esta semana lo mismo que hizo la semana pasada.
  - Para hacer un plan realista tiene que controlar su forma de gastar tiempo.
  - Para comprobar la exactitud de tus estimaciones de tiempo y planes, debe documentar y, posteriormente, comparar con lo que realmente hace.

2. La administración del tiempo

- Para gestionar su tiempo:
  - planifique su tiempo y
  - siga el plan.

2. La administración del tiempo

- 2.2 ¿Cómo utiliza su tiempo?
  - Clasifique las actividades principales :
    - 3 a 5 categorías generales, con subcategorías.
  - Registre el tiempo dedicado a cada una de las actividades principales.
  - Registre el tiempo de forma normalizada.
  - Guarde los datos de tiempo en un lugar adecuado.

2. La administración del tiempo

- 2.3 El cuaderno del Ingeniero de Software
  - Se pueden llenar varios cuadernillos. Ejemplo: uno por cada proyecto o al terminarse uno de ellos.
  - Cada uno con:
    - su portada y tiempo de inicio y fin
    - su índice
    - su lista de trabajos generales

3. Seguimiento del Tiempo

- Se debe saber establecer las tareas que interesa medir.
- El objetivo es saber el tiempo real que se está gastando
- La unidad de medida del tiempo debe ser minutos.
  - No de trabaja más de 1 hora seguida

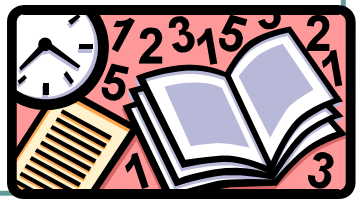
3. Bitácora de tiempo

EJEMPLO DE REGISTRO DE TIEMPOS								
Fecha	Inicio	Fin	Interrupción	A Tiempo	Actividad	Comentarios	C	U
09/09	09:00	09:50		50	Clase	Clase		
	12:40	04:18		38	Codificar	Ejercicio 1		
	14:45	15:53	10	58	Codificar	Ejercicio 1		
	18:25	07:45		80	Texto	Capítulos 1 y 2	X	2
10/09	11:06	12:19	6+5	62	Codificar	Ejercicio 1, descanso, charla	X	1
11/09	09:00	09:50		50	Clase	Clase		
	13:15	14:35	3+8	69	Codificar	Ejercicio 2, descanso, charla	X	1
	16:18	17:11	25	28	Texto	Capítulo 3, charla	X	1
12/09	18:42	21:04	10+6+12	114	Codificar	Ejercicio 3	X	1
13/09	09:00	09:50		50	Clase	Clase		
	12:38	13:16		38	Texto	Capítulo 4		
14/09	09:15	11:59	5+3+22	134	Revisión	Preparar examen, descanso, teléfono, charla		

C=Completada
U=Unidades

3.8 Ideas para su bitácora

- Traer el cuaderno todo el tiempo
- Si no se trae, anotar lo más rápido posible
- Puede ponerse hora inicial y final de interrupción
- Resumir semanalmente.



4. Planificación

- Hay dos clases de planificación:
  - Basada en período de tiempo.
  - Basada en la actividad o producto.
- Por ejemplo, leer un libro de 20 capítulos:
  - Estimar el tiempo total: 20 horas.
  - Tiempo dedicado: 1 hora a la semana.
  - Plan del producto: Leer los 20 capítulos en 20 horas.
  - Plan del período: La forma de repartir el tiempo de lectura en incrementos semanales de 1 hora.

4.2 Resumen Semanal (1)

EJEMPLO DE RESUMEN SEMANAL							Semana: 09/09		
Tarea / Fecha	Clases	Codificar	Preparar examen	Leer textos					Total
L									
M	50	98		80					223
X		95							95
J	50			71					121
V		77							77
S	50	74		40					164
D				33					33
Totales	150	339		224					713
							Número de semanas (número anterior +1) :		2

Resumen Semanal (2)

	Clases	Codificar	Preparar examen	Leer textos					Total
EJEMPLO DE RESUMEN SEMANAS ANTERIORES									
Total	150	341	134	146					771
Media	150	341	134	146					771
Máximo	150	341	134	146					771
Mínimo	150	341	134	146					771
EJEMPLO DE RESUMEN INCLUYENDO LA ULTIMA SEMANA									
Total	300	680	134	370					1484
Media	150	340	67	185					742
Máximo	150	341	134	224					771
Mínimo	150	339	0	146					713

5. Planificación del producto

- Un plan del producto adecuado requiere:
  - El tamaño y las características más importantes del producto a realizar.
  - Una estimación del tiempo requerido para hacer el trabajo.
  - Una previsión de la planificación.
- Algunas definiciones:
  - Producto: Algo que se produce para un cliente.
  - Proyecto: Produce un producto.
  - Tarea: Elemento de trabajo.
  - Proceso: Forma de hacer proyectos.
  - Plan: Forma en que un proyecto concreto va a ser hecho: cómo cuando y que costo tendrá.
  - Trabajo: Algo que hace, tanto un proyecto como una tarea.

5.7 Registro de datos de trabajos

Nombre: _____ Fecha: _____

# de trabajo	Fecha	Proceso	Estimado		Actual			Partes hasta ahora				
			Tiempo	Partes	Tiempo	Partes	Porcentaje	Tiempo	Partes	Porcentaje	Max	Min
	Descripción:											
	Descripción:											
	Descripción:											
	Descripción:											
	Descripción:											
	Descripción:											
	Descripción:											
	Descripción:											

5.7 Registro de datos de trabajos

EJEMPLO DE CUADERNO DE TRABAJOS												
Trabajo	**Fecha**	**Proceso**	**Estimado**		**Real**			**Hasta la fecha**				
			Tiempo	Unidades	Tiempo	Unidades	Tasa	Tiempo	Unidades	Tasa	Máx.	Mín.
1	09/09	Codif.	100	1	158	1	158	158	1	158	158	158
	Escribir el programa 1											
2	09/09	Texto	50	2	80	2	40	80	2	40	40	40
	Leer los capítulos 1 y 2 del libro de texto											
3	11/9	Codif.	158	1	69	1	69	227	2	114	158	69
	Escribir el programa 2											
4	12/09	Texto	40	1	28	1	28	108	3	36	40	28
	Leer el capítulo 3 del libro de texto											
5	12/09	Codif.	114	1	114	1	114	341	3	114	158	69
	Escribir el programa 3											
6	13/09	Texto	60	1	118	1	118	226	4	57	118	28
	Leer el capítulo 4 del libro de texto											

5.8 Sugerencias para registrar trabajos

- Si el trabajo es nuevo: adivinar estimado
- Si es un trabajo conocido: fijarse en estimaciones anteriores
- A la larga: usar hoja de cálculo.

6. El tamaño del producto

- La planificación del producto no es un proceso exacto.
- Para hacer un plan del producto, compare lo que planifica hacer con lo que ha hecho antes.
- Pero no todos los problemas son iguales: Base las estimaciones en problemas similares.
  - No sólo en tamaño, el tipo de problema puede variar.
- Se usará como medida las líneas de código (LOC).
- No siempre las LOC son la mejor medida.

6. Estimación del tamaño

Programa	LOC	Funciones estimadas	Mín.	Med.	Máx.
Bucles					
4	10	While sencillo			
5	14	Repeat sencillo	7	11	14
Case					
2	11	Case sencillo	5	8	11
3	14	Case grande			
Datos					
6	18	Lista enlazada sencilla			
Cálculo					
1	20	Cálculo pequeño	10	15	20
Total			22	34	45

Este programa tiene una sentencia Case sencilla, un Bucle y un cálculo. Asumo que, como máximo, el tamaño se obtendrá sumando estos tamaños típicos, 11+14+20=54 LOC. Para el valor mínimo, asumo que estas funciones podrán combinarse más efectivamente que cuando están como elementos separados. Esto nos da 22 LOC como valor mínimo. 34 LOC es el punto medio entre los dos valores anteriores.

7. Administrando su tiempo

- Revise las categorías de tiempo para ver si cubren todas sus actividades.
- Revise si son muy generales o muy detalladas.
- Para gestionar su tiempo, necesita centrarse en esas pocas categorías que consumen la mayor parte del tiempo.
- Consultando datos de semanas anteriores, puede realizar una estimación del tiempo para una nueva semana.

Una estimación de tiempo

Estudiante: Estudiante Y Fecha: 23/3/2006

Profesor: Sr. Z Clase: IP

Actividad	Minutos estimados	Minutos reales
Asistir a clase	150	
Escribir programas	360	
Leer texto	180	
Preparar exámenes	120	
Otros	30	
Total	840	

Presupuesto semanal de tiempo

EJEMPLO DE PRESUPUESTO SEMANAL DE TIEMPO (1)								Semana: 23/09
Tarea / Fecha	Clases	Codificar	Preparar examen	Leer textos				Total
L	09:00-09:50	20:30-22:30		10:20-11:00				226
M		20:30-22:30		10:20-11:00				62
X	09:00-09:50			10:20-11:00				147
J		20:30-22:30		10:20-11:00				114
V	09:00-09:50		09:00-10:00	10:20-11:00				88
S			09:00-10:00	10:20-11:00				134
D								
Totales	150	360	120	240				771

EJEMPLO DE PRESUPUESTO SEMANAL DE TIEMPO (2)		Semana : 23/09
Actividad	Minutos estimados	Minutos reales
Clase	150	
Codificar	360	
Preparar examen	120	
Leer texto	180	
Otros	30	
Total	840	

7.7 Reglas básicas de manejo del tiempo

- Gastar el tiempo como se estableció
- Las rutinas son fáciles de seguir, sobre todo si alguien las estableció.
  - Sin embargo, nosotros debemos establecer también nuestras propias reglas.
- Al hacer el presupuesto semanal, se debe agregar un “colchón” a cada actividad.

8. La gestión de los compromisos

- Un compromiso es algo que alguien espera que hagas.
- Para asegurarte de que tus compromisos son responsables y están bien gestionados:
  - Analiza el trabajo antes de aceptar el compromiso.
  - Apoya el compromiso con un plan.
  - Documenta el compromiso.
  - Si eres incapaz de cumplirlo, díselo cuanto antes a la otra parte e intenta minimizar el impacto sobre esa parte.

8. 5 Gestionar compromisos no conseguidos

- Si tiene que faltar a un compromiso, notifique inmediatamente a la otra parte, para trabajar en la resolución del problema.
- No abandones sin intentar seriamente cumplirlo:
  - Discútelo con algún experto independiente.
  - Quizás puedas añadir recursos para acelerar el trabajo.
  - Quizás puedas hacer el trabajo de una forma más “inteligente”.

8.7 Consecuencias de no gestionar compromisos

- El trabajo requerido excede el tiempo disponible.
- Fallar al enfrentarte a los compromisos.
- Prioridades mal colocadas.
- Pobre calidad del trabajo.
- Pérdida de confianza.
- Pérdida de respeto a tus opiniones.

Tabla de compromisos

Ejemplo de Compromisos				
Fecha comprometida	**Compromiso**	**¿Con quién?**	**Horas**	**Consigo**
Semanal				
L, M y V	Asistir a clase	Profesor	1,5	Aprobar
L, M y V	Entregar trabajo informal	Profesor	6	Aprobar
M y J	Leer libro	Profesor	4	Aprobar
L, M , X , J y V	Trabajo tiempo parcial	Admisión	10	Paga
Otros				
	Ejercicio trimestral	Profesor	24	Aprobar

9. Administración de Calendarios

- El diagrama de Gantt.
  - Identifica con bastante detalle las distintas tareas que componen el trabajo.
  - Estima el tamaño para cada una de pequeñas tareas y determina la cantidad de trabajo que probablemente necesitarán.
  - Registra cada tarea en el diagrama de Gantt con una barra.

9. Administración de Calendarios

- Además:
  - Asegurarse de que cada individuo conoce las tareas que tiene que hacer.
  - Obtener un compromiso de fechas para cada una de estas tareas.
  - Identifica las interdependencias entre las tareas y documéntalas.
  - Revisa la programación propuesta y las interdependencias con todas las personas implicadas.
  - Revisa la programación para asegurarte que cubre todas las tareas necesarias para completar el trabajo.

9. 4 Puntos de control (1)

- Cuando se completa cada parte, se ha realizado un determinado grado de progreso.
- Estos puntos de la programación que son medibles se llaman puntos de control o hitos.
- Un hito es un punto que, objetivamente, se puede identificar en un proyecto.
- Para ser útiles deben ser claros y no ambiguos.
- 2 hitos por semana, aproximadamente.

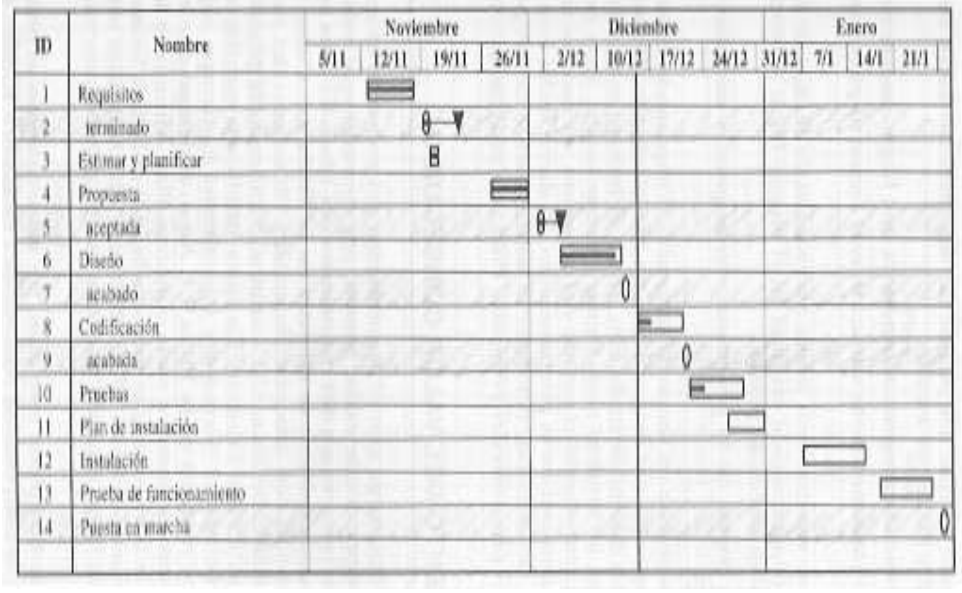
9.4 Puntos de control (2)

- Ejemplos buenos:
  - Elaborado y documentado el plan para escribir el programa, utilizando un formato normalizado.
  - Completado y documentado un diseño de un programa, con un formato normalizado.
  - Implementado, compilado y corregido un programa.
- Ejemplos malos:
  - Finalizado un plan para escribir un programa.
  - Diseñado un programa.
  - Completado el 90% de la codificación.

9.4 Puntos de control (3)

- El seguimiento de un plan permite determinar si el proyecto va adelantado o retrasado.
  - Informar sobre el estado real es esencial cuando los proyectos se hacen para los clientes, que son los que pagan (y los jefes).

Ejemplo de Diagrama de Gantt



11. El proceso de desarrollo de Software (1)

- Un proceso es un conjunto definido de pasos para hacer un trabajo.
  - Cada paso o fase de un trabajo tiene especificados unos criterios de entrada que deben ser satisfechos antes de comenzar la fase.
  - Cada fase tiene unos criterios de salida que deben satisfacerse antes de terminar la fase.
  - Sin dichos datos, no hay forma de decirles si van mejorando o empeorando
- El PSP es un marco de trabajo que ayuda a los ingenieros de software a medir y mejorar su forma de trabajar.

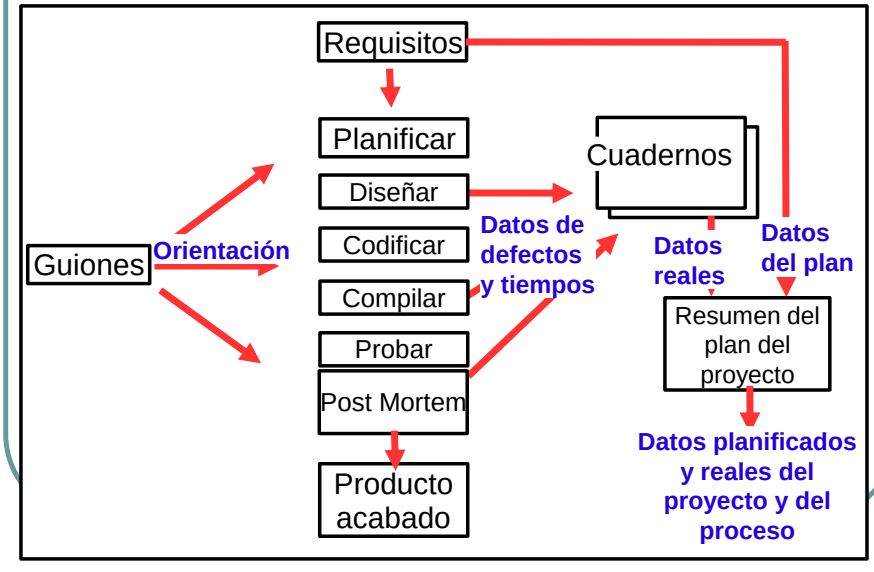
Algunas definiciones (1)

- Producto:
  - algo que produces para un colaborador, un empresario o un cliente.
- Proyecto:
  - normalmente produce un producto.
- Tarea:
  - Elemento de trabajo.
- Proceso:
  - define la forma de hacer proyectos.
  - tienen varias fases o pasos:
    - planificación, desarrollo y pruebas.
    - Una fase puede estar compuesta de tareas o actividades.

Algunas definiciones (2)

- Los planes describen la forma en que un proyecto concreto va a ser hecho: cómo, cuándo y qué coste tendrá.
- Cuando un proceso esta totalmente descrito, se denomina *proceso definido*.
  - Están compuestos normalmente de guiones, tablas, plantillas y estándares.
- Guión del proceso:
  - Conjunto de pasos escritos, que los usuarios o agentes del proceso siguen cuando utilizan el proceso.

El proceso de desarrollo de SW (y 2)



Puntos de Control y Fases

- Los puntos de control ayudan a hacer y controlar las programaciones de los proyectos.
  - Definiendo de forma explícita y clara los puntos de control del proyecto,
  - puntos de control proporcionan puntos de referencia precisos.
  - para medir el estado del proyecto mientras se está haciendo el trabajo.
- Con un proceso definido, cada fase produce un resultado específico y por lo tanto la conclusión de una fase es un punto de control medible.

El guión del proceso

- Planificación. Análisis, requisitos.
- Diseño.
- Codificación.
- Compilación y corrección de errores.
- Pruebas.
- Post mortem. Se definen las tareas finales que hay que realizar para asegurar que el trabajo ha sido terminado.

El resumen del plan del proyecto

- Describe el proceso básico del PSP y muestra como un proceso definido, puede ayudar a mejorar tus planes.
- La tabla del Resumen del Plan del Proyecto aumenta para incluir:
  - los tiempos de las fases del proyecto
  - calcular el tiempo hasta la Fecha
  - el porcentaje del tiempo de desarrollo dedicado a cada fase.
- Haciendo planes de proyectos:
  - Se podrá estimar el tiempo que se dedica a cada fase.
  - Basada en experiencias anteriores, utilizando para ello los valores de % Hasta la Fecha de los programas anteriores.

12. Defectos

- El término BUG parece que se refiere a cosas malditas que deben ser aplastadas o ignoradas, lo cual trivializa el problema.
- Si se llamaran Bombas de Efecto Retardado, ¿sentiría la misma sensación de alivio cuando supieras que tras probar un programa sólo quedan unas pocas?

12. Defectos

- Un defecto es algo OBJETIVO que está equivocado en un programa:
  - Error sintáctico, falta tipográfica, error de puntuación, ...
- Pueden estar en los programas, en los diseños o incluso en los requisitos.
- Los errores causan defectos, y todos provienen de errores humanos.
- Es decir, las personas cometen errores y los programas tienen defectos.

12.5 Tipos de defectos

- Lista procedente del trabajo de Chillagere y sus colegas en el centro de investigación de IBM:

Tipos de defectos		
Nº de tipo	Nombre del tipo	Descripción
10	Documentación	Comentarios, mensajes
20	Sintáxis	Ortografía, puntuación, erratras, formato de las instrucciones
30	Construir, paquetes	Gestión del cambio, librerías, control de versión
40	Asignación	Declaración, nombres duplicados, ámbito, límites
50	Interfaz	Llamadas a procedimientos y referencias, E/S, formatos de usuario
60	Chequeo	Mensajes de error, chequeos inadecuados
70	Datos	Estructura, contenido
80	Función	Lógica, punteros, bucles, recursión, computación, defectos de la función
90	Sistema	Configuración, temporización, memoria
100	Entorno	Diseño, compilación, pruebas y otros problemas que soporta el sistema

Gestión de los defectos

- Registra cada defecto que encuentres en un programa.
- Registra la información suficiente sobre cada defecto para que puedas entenderlo posteriormente.
- Analiza estos datos para ver qué tipos de defectos causan los mayores problemas.
- Idea formas de encontrar y corregir estos defectos.

12.10 Gestión de los defectos

Ejemplo de Registro de Defectos						
Fecha	Número	Tipo	Introducido	Eliminado	Tiempo de correción	Defecto corregido
28/10/04	1	20	Codificación	Compilación	1	
Descripción: Omitido ;						
	2	20	Codificación	Compilación	1	
Descripción: Omitido ;						
	3	40	Diseño	Compilación	1	
Descripción: Defecto en la parte derecha del operador binario, debe tratarse el entero como float						
	4	40	Codificación	Compilación	1	
Descripción: Error en la parte derecha, la constante debería ser 0,0 y no 0						
	5	40	Diseño	Compilación	7	
Descripción: El exponente debe ser un entero, investigué y utilizé la librería matemática para sqrt, la integral no se calculó correctamente						
	6	80	Codific	Pruebas	14	
Descripción: El bucle no terminó con un exponente negativo, olvidó cambiar el signo en la sustracción						

13. Calidad del Software

- Afecta a los costes de desarrollo, programación de entregas y satisfacción del usuario.

- ¿Otras definiciones?

13.2 Encontrar defectos

- Aunque no hay forma de acabar con la introducción de defectos, es posible encontrar y eliminar casi todos los defectos al principio del desarrollo.
- Siempre están implicados estos métodos:
  - Identificar los síntomas del defecto.
  - Deducir de estos síntomas la localización del defecto.
  - Entender lo que es erróneo en el programa.
  - Decidir cómo corregir el defecto
  - Hacer la corrección.
  - Verificar que el arreglo ha resuelto el programa.

13.3 Formas de encontrar defectos (1)

- Con el compilador.
  - Pero no detecta los errores semánticos.
- Mediante pruebas.
  - Las pruebas de unidad encuentra sobre el 50% de los defectos lógicos.
- Las de sistema entre un 30% y un 40%. Pero no podemos probar todos los casos.
- La más común de todas: Que los detecten los usuarios.
  - Durante un año, IBM gastó 250 millones de dólares en reparar y reinstalar correcciones de 13,000 errores encontrados por los usuarios: 20,000 dólares por defecto.

13.3 Formas de encontrar defectos (2)

- Según Humphrey, la forma más rápida y eficiente es revisando personalmente el código fuente.
- Así se ven los problemas, no los síntomas.
- Sin embargo, con experiencia encontrará una media del 75% al 80% de los defectos.
- Se necesitan, al menos, 30 minutos para revisar 100 LOC.

13.5 ¿Por qué hay que encontrar pronto los errores?

- Imagina que vas a comprar un coche, y visitas 2 fábricas.
- En la 1ª encuentran una media de 10 defectos por coche en las pruebas de los coches, que son corregidos antes de enviar el coche al concesionario.
- En la 2ª encuentran 1 defecto por cada 10 coches. El resto lo encuentran los compradores.

13.6 Coste de encontrar y corregir errores (1)

- Durante la revisión, se encuentra 1 error cada 1 ó 2 minutos.
- Durante las pruebas de unidad, 1 error cada 10 ó 20 minutos.
- En las pruebas de integración, 10 a 40 horas.

13.6 Coste de encontrar y corregir errores (2)

- Datos reales:
  - Una pequeña empresa:
    - Con PSP, las pruebas de integración duraron 2 semanas.
    - Con el módulo desarrollado sin PSP, las pruebas duraron varias semanas, con 300 horas por defecto.
  - Un sistema aeroespacial necesitó:
    - una media de 40 horas por defecto en las pruebas del sistema de navegación aérea.
  - En Digital Equipment Corporation,
    - para un sistema, el tiempo mínimo para encontrar y corregir cada defecto informado por el cliente fue de 88 horas.

13.8 Revisar antes de compilar

- Dedicarás el mismo tiempo antes o después de compilar.
- Antes de la revisión, dedicarás entre un 12% y un 15% del tiempo a compilar. Después un 3% o menos.
- Una vez compilado el programa, la revisión no es tan completa.

13.8 Revisar antes de compilar

- La compilación es igualmente efectiva antes o después de la revisión del código.
- La experiencia indica que cuando un programa tiene muchos defectos durante la compilación, generalmente tienen muchos defectos en las pruebas.

14. Listas de comprobación (1)

- La clave para realizar una revisión de código efectiva es tener un procedimiento de revisión eficiente.
- Una lista de comprobación contiene una serie de pasos de procedimiento que quieres seguir de forma precisa.

14. Listas de comprobación (1)

- Un ejemplo de lista de comprobación completa y compleja es la que realiza la NASA en la cuenta atrás de un lanzamiento, que dura varios días.
- La lista de comprobación encapsula la experiencia personal.
  - Utilizándola con regularidad y adaptándola, permitirá la detección oportuna de los defectos de los programas.

14. Listas de comprobación (2)

- El principal peligro es que generalmente encuentra lo que busca.
  - Si sólamente hace las pruebas de la lista de comprobación, sólamente encontrará lo que está en dicha lista.
- Haga al menos una revisión general del programa para buscar lo inesperado, desde la perspectiva del sistema o del usuario.

Método para llenar la lista de comprobación de ejemplo

- Cuando completes cada paso de la revisión, anota el número de defectos que has encontrado de cada tipo en la casilla de la derecha. Si no hay ninguno, anota un control en la casilla de la derecha.
- Completa la lista de comprobación para un programa, clase, objeto o método antes de comenzar a revisar la siguiente.

14.2 Ejemplo de lista de comprobación (1)

Propósito	Guía	#	#	#	#	Hasta la fecha	% Hasta la fecha
Completo	Verifica que todas las funciones del diseño están programadas	X					
Includes	Verifica que las sentencias import están completas	X					
Inicio	Comprobar la inicialización de parámetros y variables: • Al inicio del programa. • Al comenzar cada bucle. • En la entrada a un procedimiento o función.	X					
Llamadas	Comprobar los formatos de las						

14.4 Clasificación de datos de defectos

Ejemplo de Análisis de Errores							
Tipo	Introducido			Eliminado			Omitido
	Diseñar	Codificar	Otros	Revisar	Compilar	Pruebas	En revisi
10							
20		8		4	4		4
30	2	3		1	4		4
40		2			1	1	2
50							
60							
70							
80	2	3			1	4	5
90							
100							
Total	4	16		5	10	5	15
Programa							
10	2	6			6	2	8
11	1	5		3	2	1	3
12	1	5		2	2	2	4

15.5 Estimación de defectos

- Un Ingeniero de Software experimentado introduce entre 50 y 250 defectos/KLOC.
- Para calcular el total de defectos por KLOC (Dd) en cada programa:

$$Dd = 1000 * D/N$$

- (D = Defectos encontrados, N = Líneas de código nuevas o cambiadas)

15.5 Estimación de defectos

- Estima el número de LOC del nuevo programa.
- Calcula el valor medio de defectos/KLOC de los programas anteriores.
- Dd = 1000 * (D1+...+Di) / (N1+...+Ni)

Nº de programa	Defectos (D)	LOC
1	6	37
2	11	62
3	7	49
4	9	53
5	5	28
Total hasta la fecha	38	229

16. La economía de eliminar defectos

- El software de las primeras impresoras láser, tenían unas 20,000 LOC, actualmente entorno a 1,000,000 LOC.
- Los coches actuales, tienen software con varios miles de LOC.
- En MS, 250 ingenieros del sistema NT dedicaron 1 año completo a encontrar y depurar 30,000 defectos:
  - 16 horas por defecto.

Consejos

- Registrar todos los defectos.
- Hacer mejores modelos, más completos y mejor documentados.
- Utiliza los mejores métodos.
- Utiliza las mejores herramientas.

17. Defectos de diseño

- ¿Qué contabilizamos como defectos de diseño?
  - Los defectos introducidos en la fase de diseño
  - Aquellos tipos de defectos que implican cuestiones de funciones de codificación, lógica, rendimiento y sincronización.

17.5 Causas de los defectos de diseño

- Decisiones de diseño incorrectas.
- Tomando la decisión de diseño correcta, comete un error.
  - Ejemplo: si no incluye todos los casos de ejecución de un bucle.
- Problema de interpretación literal: Se comprenden los requisitos pero no se entiende el contexto.
  - Ejemplo: Construye el procedimiento incorrecto.

18. Calidad del producto

- Las pruebas son caras, aunque sea para pequeños programas.
- Cuanto más complejo es el producto, las pruebas consumen más tiempo y son más caras.
- También será más costoso encontrar y corregir cada defecto

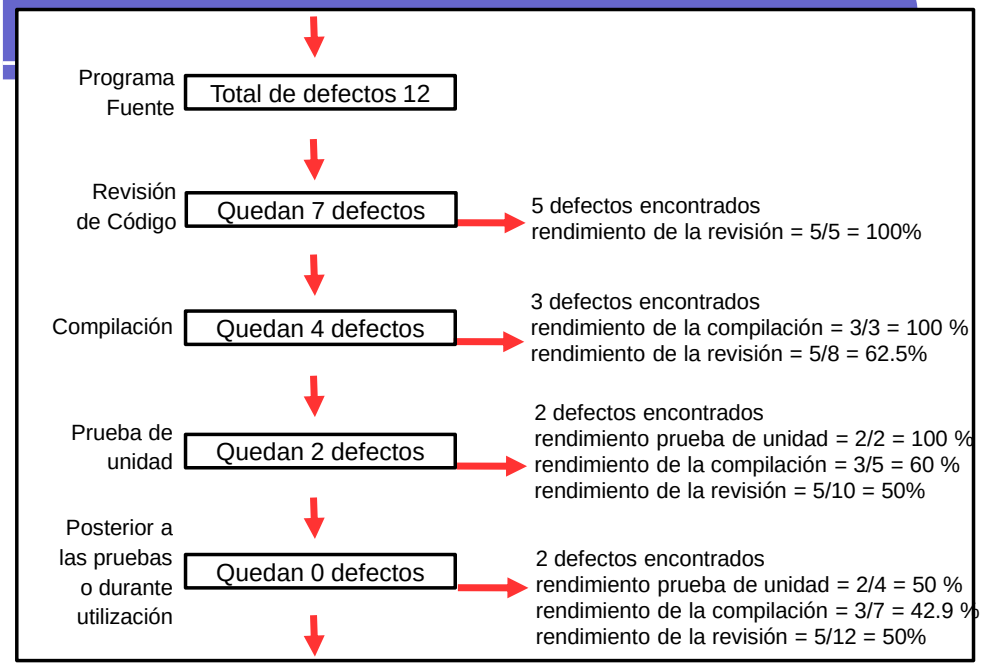
Dificultad de encontrar errores (1)

- Los defectos enmascaran o agravan a otros.
  - Interaccionan y enmascaran síntomas de otros.
- Es difícil, incluso en programas pequeños, probar todos los caminos lógicos.

Dificultad de encontrar errores (2)

- En sistemas complejos, al probar sólo las condiciones que pensamos más importantes, pasamos por alto muchos defectos.
- A mayor número de defectos que entran en la fase de pruebas, compilación o revisión, mayor la probabilidad de dejarlos en el producto.

18.5 Valores de rendimiento



19. Calidad del proceso

- La medida fundamental de un proceso tiene que ver con el volumen de productos realizados, su calidad, el tiempo y los recursos requeridos para hacer el trabajo.
- La tasa de eliminación de defectos disminuyen conforme mejora la calidad del producto.

19.3 Una estrategia para la eliminación de errores (1)

- Esforzarse en desarrollar módulos con la máxima calidad posible.
- Hacer inspecciones de todas las interfaces de módulos y sus interacciones.
- Inspeccionar los requisitos para asegurarte que todas las funciones importantes son adecuadamente entendidas, diseñadas e implementadas.

19.3 Una estrategia para la eliminación de errores (2)

- Inspeccionar el sistema y el diseño del programa frente a los requisitos, para asegurar que son tratados adecuadamente todos los requisitos clave.
- Hacer unas pruebas de unidad exhaustivas después de que se haya inspeccionado el código.
- Hacer una prueba de integración global.
- Hacer pruebas a todo el sistema.

19.4 El costo de la calidad (1)

- Como Ingenieros de Software necesitamos un equilibrio entre el tiempo dedicado y la calidad de los productos hechos.
- El Coste de la Calidad (CDC) proporciona una forma de tratar estas cuestiones.
- Tiene 3 elementos principales: Costes de los fallos, costes de valoración y costes de prevención.

19.4 El costo de la calidad (2)

- Los costes de los fallos incluyen todos los costes de corregir los defectos del producto: Corregir defectos, re-diseñar, re-compilar y re-probar.
- Los costes de valoración incluyen todo el trabajo de valoración del producto para ver si tiene defectos, excluyendo el tiempo dedicado a la corrección de defectos.
- Los costes de prevención son los costes incurridos cuando modificas el proceso para evitar introducir errores: Análisis para comprender los defectos, mejora de especificación de requisitos, diseño e implementación, rediseño y pruebas de un nuevo proceso.

Resumen del plan del proyecto (1)

Resumen	Plan	Real	Hasta la fecha
Minutos/LOC	5,48	4,6	5,35
LOC/Hora	10,95	13,04	11,21
Defectos/KLOC	92,53	52,6	86,7
Rendimiento	40	100	45,5
V/F	0,38	1,93	0,44

Tamaño programa (LOC)	Plan	Real	Hasta la fecha
Total nuevo & cambiado	49	57	392
Tamaño máximo	62		
Tamaño mínimo	36		

Tiempo por Fase (min.)	Plan	Real	Hasta la fecha	%Hasta la fecha
Planificación	17	20	140	6,7
Diseño	29	38	253	11,1
Codificación	116	119	911	43,4
Revisión del código	21	29	174	8,3
Compilación	15	5	105	5
Pruebas	41	10	289	13,8
Postmorten	30	41	247	11,7
Total	269	262	2099	100
Tiempo máximo	340			
Tiempo mínimo	197			

Resumen del plan del proyecto (y 2)

Defectos Introducidos	Plan	Actual	Hasta la fecha	%Hasta la fecha	Def./Hora
Planificación					
Diseño	1		5	14,7	1,29
Codificación	4	3	28	82,4	1,84
Revisión del código					
Compilación			1	2,9	
Pruebas					
Total	5	3	34	100	

Defectos eliminados	Plan	Actual	Hasta la fecha	%Hasta la fecha	Def./Hora
Planificación					
Diseño					
Codificación					
Revisión del código	2	3	15	44,1	5,17
Compilación	2		13	38,2	7,43
Pruebas	1		6	17,1	1,25
Total	5	3	34	100	

Guión del proceso PSP (1)

Guión del proceso PSP	
Entradas requeridas	• La descripción del problema. • Tabla Resumen del Plan del Proyecto PSP. • Una copia de la lista de comprobación para la revisión de código. • Datos de tamaños y tiempos reales de programas anteriores. • Cuaderno de Registro de tiempos. • Cuaderno de Registro de Defectos
1 Planificación	• Obtén una descripción de las funciones del programa. • Estima las LOC máx., mín., total requeridas. • Determina los minutos/LOC. • Calcula los tiempos de desarrollo máx., mín. y total. - Estima los defectos a introducir y eliminar en cada fase. • Estima los defectos a introducir y eliminar en cada fase. • Escribe los datos del plan en la tabla Resumen del Plan del Proyecto. • Anota el tiempo de planificación en el Cuaderno de Registro de Tiempos.
2 Diseño	• Diseña el programa. • Anota el diseño en el formato especificado. • Anota el tiempo de diseño en el Cuaderno de Registro de Tiempos.
3 Codificación	• Implementa el diseño. • Utiliza un formato estándar para introducir el código. • Anota el tiempo de codificación en el Cuadero de Registro de Tiempos.
4 Revisión de código	• Revisar completamente el código fuente. • Seguir el guión de revisión de códig de la lista de comprobación. • Corregir y registrar todos los defectos encontrados. • Registrar el tiemop de revisión en el Cuaderno de Registro de Tiempos.

Guión del proceso PSP (2)

5 Compilación	• Compila el programa. • Corrige y registra todos los errores encontrados. • Anota el tiempo de revisión en el Cuaderno de Registro de Tiempos.
6 Pruebas	• Prueba el programa. • Corrige y registra todos los errores encontrados. • Anota el tiempo de revisión en el Cuaderno de Registro de Tiempos.
7 Postmorten	• Corrige y registra todos los errores encontrados.Completa la tabla Resumen del Plan del Proyecto con los datos de tiempo, tamaño y defectos reales. • Revisa los datos de defectos y actualiza la lista de comprobación para la revisión de código. • Anota el tiempo postmortem en el Cuaderno de Registro de Tiempos.
Criterios de salida	• Programa probado a fondo. • Diseño adecuadamente documentado. • Lista de comprobación para la revisión de código completa. • Listao completo del programa. • Resumen del Plan del Proyecto completo. • Cuaderno de Registro de tiempos y defectos completos.

20. Un compromiso personal con la calidad

- Cuando el software forme parte de un sistema de vuelo de aviones, de conducción de coches, de gestión de tráfico aéreo, de funcionamiento de una fábrica, control de plantas nucleares...

Sus defectos tendrían consecuencias peligrosas.