

Una triste historia de errores en programación ...

El Mito del Primer *Bug* (1/3) (Harvard, 1945)

92.

9/9

0800 Antan started
1000 " stopped - antan ✓
13⁰⁰ MC 1032 HP-MC 2.130476415 (2) 4.615925059 (-2)
(033) PRO-2 2.130476415
convd 2.130676415
Relays 6-2 in 033 failed speed test
in Relay 11.00 test.
Relays changed

1100 Started Cosine Tape (Sine check)
1525 Started Multi Adder Test.


1545 Relay #70 Panel F
(moth) in relay.

First actual case of bug being found.
1700 closed down.

Una palomilla, atascada en los circuitos (relés), bloqueó el funcionamiento de la computadora Mark II (¿Puede imaginar qué calculaba esta computadora?)

La matemática Grace Murray Hopper ha decidido entonces llamar a todo lo que interrumpe el funcionamiento normal de un programa con la palabra *BUG*:

1545



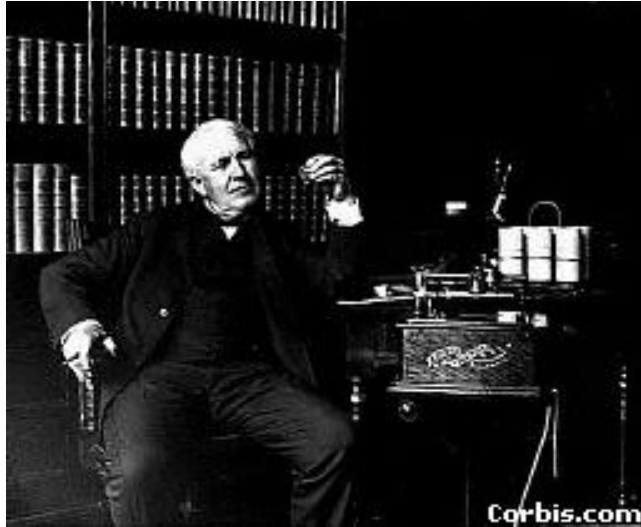
Relay #70 Panel F
(moth) in relay.

First actual case of bug being found.
anchant started.

~~163~~ 1630

El Mito del Primer *Bug* (2/3)

(Harvard, 1945)



Hay que decir que el término “*bug*” ya se usaba antes:

Thomas Alva Edison, por ejemplo, usó este término en una carta (de 18/11/1878) donde él explica el proceso de trabajo con sus inventos.

Histoire de l'Informatique, <http://histoire.info.online.fr>
Fred R. Shapiro, *The First Bug*,
<http://byte.com/art/9404/sec15/art1.htm>

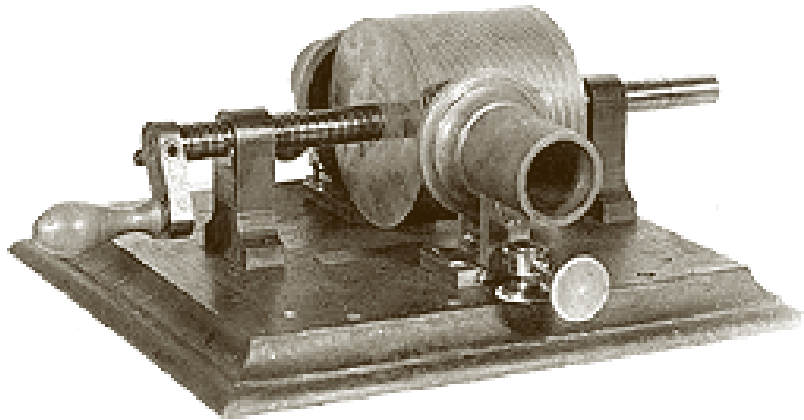
bug.

b. A defect or fault in a machine, plan, or the like. orig. U.S.

1889 Pall Mall Gaz. 11 Mar. 1/1 Mr. Edison, I was informed, had been up the two previous nights discovering ‘a bug’ in his phonograph - an expression for solving a difficulty, and implying that some imaginary insect has secreted itself inside and is causing all the trouble.

Oxford English Dictionary, 2nd Edition

El Mito del Primer *Bug* (3/3) (Harvard, 1945)



Phonograph of T.A. Edison (1877)
(the first sound recording is here)

Mary had a little lamb,
Its fleece was white as snow;
And everywhere that Mary went
The lamb was sure to go.

Mother Goose Rhymes

Errores de programación famosos tristemente (1/20)

¿Cuánto cuesta una coma?

En FORTRAN el siguiente bucle usa la variable *I* como el índice de bucle para ejecutar su cuerpo cinco veces. La sentencia “20 CONTINUE” marca el fin del cuerpo del bucle. Pero hay un error en uso del punto en vez de la coma necesaria entre el 1 y el 5:

```
DO 20 I = 1.5  
      .  
      .  
      .  
20    CONTINUE
```

Como FORTRAN generalmente ignora espacios, y las variables no declaradas que comienzan con la letra “*D*” se asumen como *REAL*, entonces el reemplazo del “,” por el “.” causa que la sentencia DO se interpreta como la sentencia de asignación:

```
DO20I = 1.5
```

Errores de programación famosos tristemente (2/20)

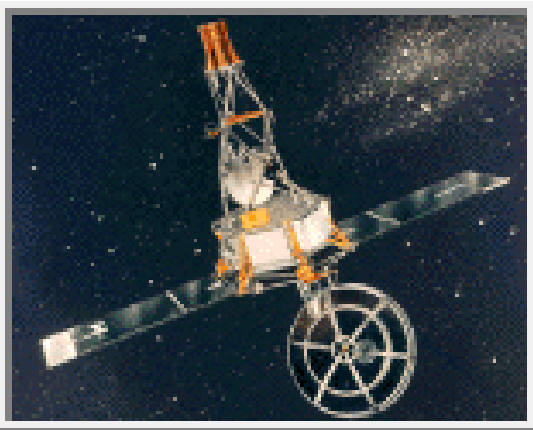
¿Cuánto cuesta una coma? 80 millones de dólares...

donde *DO20* es una variable de tipo *REAL*. La sentencia

20 CONTINUE

no parece extraña a FORTRAN, imposible en C o PASCAL.

Este escenario provocó la pérdida de la primera nave norteamericana a Venus, **Mariner 1**, en 22 de julio de 1962, que terminó su viaje en el Atlántico (80 millones de dólares).



(Phillip A. Laplante, *Real-Time Systems Design and Analysis. An Engineer's Handbook*, pp.70-71)

Mariner 2: the world's first successful interplanetary spacecraft (27/08/1962)

Errores de programación famosos tristemente (3/20)

FIFO, LIFO & GIGO

FIFO, LIFO y

GIGO (*Garbage-Input-Garbage-Output*) – “Si la basura entra, la basura sale”:

Un cohete fue programado para regresar al mismo punto de despegue en la superficie terrestre exactamente 24 horas después de lanzamiento, cuando la tierra hubiera hecho una rotación completa. Falla por cerca de 100 millas.

Un error humano: la tierra no hace su rotación en 24 horas exactas.

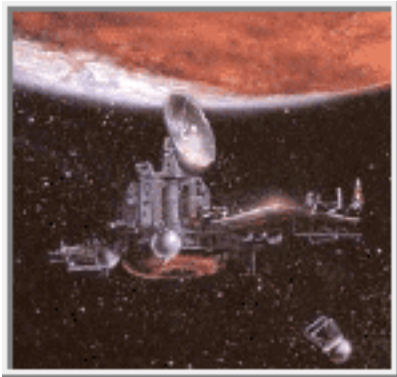
(Francis Scheid, *Introducción a la Ciencia de las Computadoras*, p.138)

La cápsula tripulada de la nave **Gemini V** se desvió del punto de aterrizaje a 100 millas porque su programa de navegación ignoró el movimiento de la tierra alrededor del Sol.

(H.Lin, *The development of software for ballistic-missile defense*, *Scientific American*, vol.253, no.6 (Dec.1985), p.49)

Errores de programación famosos tristemente (4/20)

Phobos 1, Apollo 8, Apollo 14



La nave soviética Phobos 1 al Marte fue perdida por la errónea corrección del software (30/08/1988), lo que costó 300 millones de rublos. Su desorientación rompió el enlace de radio y las baterías solares se descargaron.

(Aviation Week, 13 de febrero de 1989)



Una parte de la memoria de la computadora de la nave espacial **Apollo 8** fue borrada.



Durante el vuelo de 10 días del **Apollo 14** fueron detectados 18 errores.

(G.J.Myers, Software Reliability: Principles & Practice, p.25)

Errores de programación famosos tristemente (5/20)

Desbordamiento y excepciones

El 19 de septiembre de 1989 un **desbordamiento** (*overflow*) de un entero de 2 bytes provocó el colapso de la computadora del **hospital en Washington, DC**. Lo que obligó a pasar todas las actividades del hospital a modo manual.



Un avión caza F-18 cayó por no procesar la condición de excepción.

(ACM SIGSOFT Software Engineering Notes, vol.9, no.5)

Errores de programación famosos tristemente (6/20)

Caso de Patriot



El 25 de febrero de 1991 durante la Guerra de Golfo una batería norteamericana de misiles Patriot en Dharan, Arabia Saudita, falló en interceptar un misil iraquí Scud.

Murieron 28 soldados. La causa fue el cálculo inexacto del tiempo.

El reloj interno del sistema medía tiempo en las décimas de segundo. El tiempo en segundos se obtenía multiplicando el valor del contador por 1/10. Este cálculo se realizaba en un registro de 24 bits.

Caso de Patriot

$$0.1_{10} = 0.000110011001100110011001100110011001100_2 \dots$$

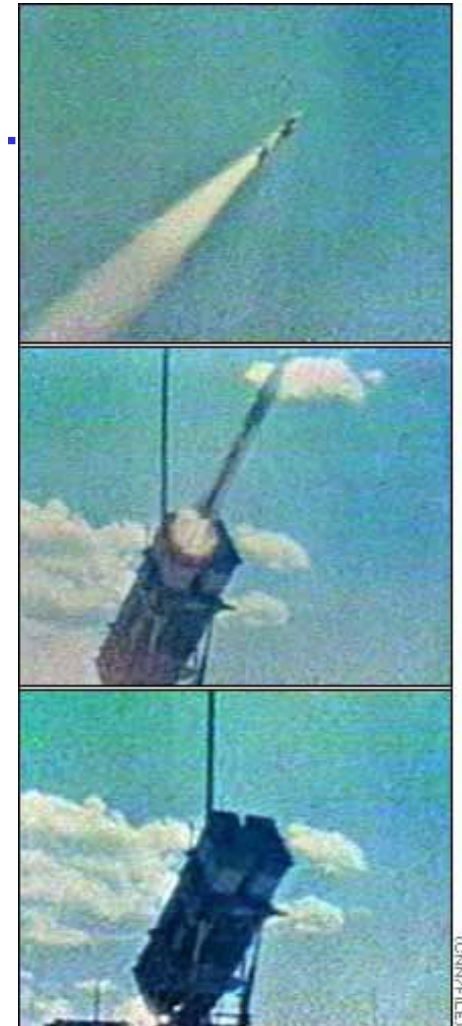
Reg = 0.00011001100110011001100

$$\begin{aligned} \text{Err} &= 0.0000000000000000000000000110011001100\dots \\ &\approx 0.00000095_{10} \end{aligned}$$

0.000000095*100*60*60*10 = 0.34 seg

$$1,676 \text{ m/seg} * 0.34 \text{ seg} = 569.84 \text{ m}$$

**(GAO/IMTEC-92-26, Patriot Missile Defense:
Software Problem Led to System Failure
at Dhahran, Saudi Arabia)**



Errores de programación famosos tristemente (8/20)

Caso de Ariane 5



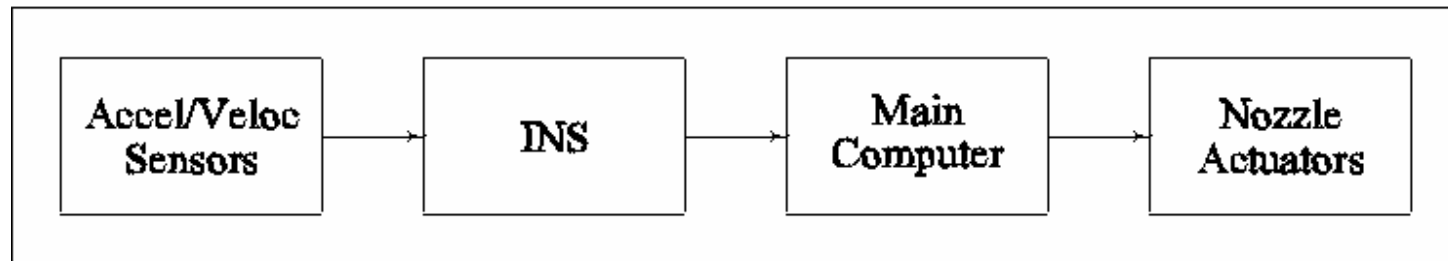
La mañana del 4 de junio de 1996 fue parcialmente nublada en Kourou, Guyana, cuando la Agencia Espacial Europea (ESA) preparaba el primer lanzamiento del cohete francés Ariane 5.

El cohete despegó a las 9:34.

Solamente 37 segundos después el cohete viró brusca-mente y comenzó destruirse. El mecanismo de la seguridad identificó la catástrofe inminente y inició la secuencia de autodestrucción. El accidente fue provocado por el error de software. De \$7 billones y un año de retraso del programa espacial.

Errores de programación famosos tristemente (9/20)

Caso de Ariane 5



INS – Inertial Navigation System

H0 = 09h 33mn 59s

La función de alineamiento es operativa durante 50 seg después del inicio *Flight Mode* de los SRIs (Sistemas de Referencia Inercial 1 y 2) que ocurrió en H0 – 3 seg. Por consiguiente, después del despegue la función fue operativa durante aproximadamente 40 seg del vuelo. Esto fue el requerimiento para el Ariane 4 y no fue necesario para el Ariane 5.

Errores de programación famosos tristemente (10/20)

Caso de Ariane 5

El **Error de Operando** sucedió por un alto valor inesperado de la función interna de alineamiento llamado BH (*Horizontal Bias* – Sesgo Horizontal) relacionado a la velocidad horizontal.

El valor de BH más grande que el esperado sucedió porque la parte inicial de la trayectoria del Ariane 5 es diferente de la trayectoria del Ariane 4 e implica valores de velocidad horizontal considerablemente más altos.

Pero el software del Ariane 4 fue dejado intacto para el Ariane 5.

Errores de programación famosos tristemente (11/20)

Caso de Ariane 5

El error ocurrió en la parte del software que calculaba los resultados significantes solamente para los momentos antes del despegue.

La excepción del software interno de SRI sucedió durante la conversión de datos del formato en punto flotante de 64 bits al valor entero con signo de 16 bits.

Errores de programación famosos tristemente (12/20)

Caso de Ariane 5

Un ejemplo de simulación no en Ada sino en Pascal:

```
program Ariane_5;

var  X : Real;      {      2.9e-39 .. 1.7e38      }
     Y : Longint;   { -2147483648 .. 2147483647 }

begin
  X := 215e7;
  Y := Round(X); { En Ada es:  y = int(x) }
end.

{ Error 207: Invalid floating point operation. }
```


Errores de programación famosos tristemente (13/20)

Caso de Ariane 5

El resultado fue el *Error de Operando*. Esta conversión de datos no fue protegida como las otras cuatro conversiones.

El SRI 1 se apagó y en 72 ms transfirió el control al sistema de respaldo SPI 2. El SRI 2 obtuvo el mismo error y también se apagó transfiriendo a su vez el control a la computadora del bordo OBC (*On-Board Computer*) y pasándole, en vez de datos, el patrón diagnóstico de bits que finalmente fue interpretado como los datos reales del vuelo.

Errores de programación famosos tristemente (14/20)

Caso de Ariane 5



Según este dato la computadora principal cambió el ángulo de ataque del cohete a más de 20 grados.

Bajo las altas fuerzas aerodinámicas sucedió la separación de la primera etapa del cohete y fue activado el sistema de autodestrucción en $H0 + 39$ seg.

(Ariane 5. Flight 501 Failure Report by the Inquiry Board.

M.Ben-Ari. The Bug That Destroyed a Rocket. . . .)

Errores de programación famosos tristemente (15/20)

Caso de Therac-25

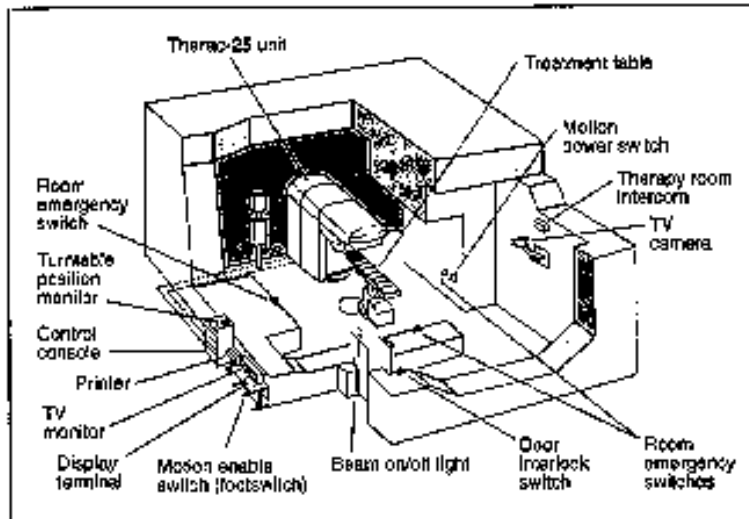


Figure 1. Typical Therac-25 facility.

Entre junio de 1985 y enero 1987 se conocen 6 accidentes por la sobredosis masiva en la máquina computarizada de la terapia rayos X Therac-25.

**El error de software fue producido por la condición de concurso entre las tareas concurrentes.
Dos personas murieron...
Muchos se quedaron heridos...**

Errores de programación famosos tristemente (16/20)

US Vincennes y A320

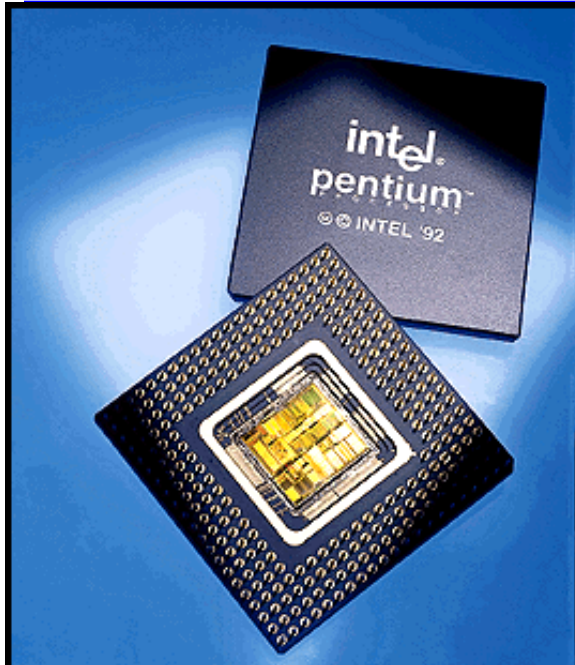


En 1988 una nave militar de EE.UU. lanza un misil y derriba el avión civil Airbus A320 al confundirlo con un F-14.

Error de software. Murieron 290 personas inocentes...

Errores de programación famosos tristemente (17/20)

Caso del procesador Pentium



“June 1994: Intel testers discover a division error in the Pentium chip. Intel managers decide that the error will not affect many people and do not inform anyone outside the company. This was Intel's first mistake...”

En diciembre de 1994, *Intel Corporation* reemplaza gratis todos los chips Pentium por tener un pequeño error en el circuito de división de números reales.

El error fue detectado en el sexto dígito decimal del resultado de división de algunos números. Por ejemplo, $81919.9968 / 1.874999925$ daba como resultado **43690.53** en vez de **43690.6667**.

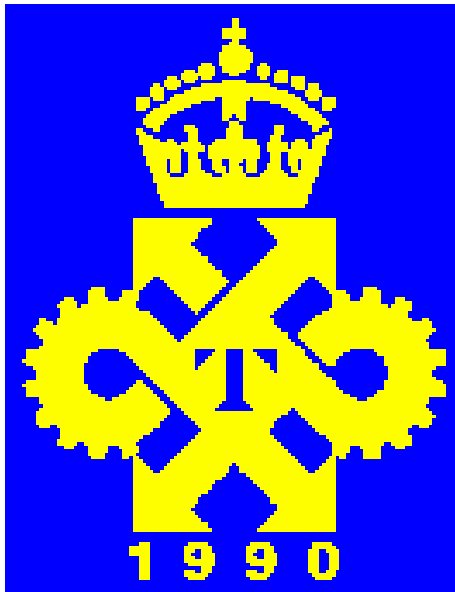
El costo del error fue evaluado a casi 400 millones de dólares.

La causa: el diseño no correspondía a la especificación.

Errores de programación famosos tristemente (18/20)

El caso contrario

En 1984 la empresa británica *Inmos* construye su transputer, primera microcomputadora de diseño británico en un solo chip. Bill Roscoe de Oxford construye una demostración matemática de la corrección del diseño. En dos semanas fueron descubiertos tres errores. Al corregirlos **fue probado matemáticamente que el diseño corresponde a su especificación.**

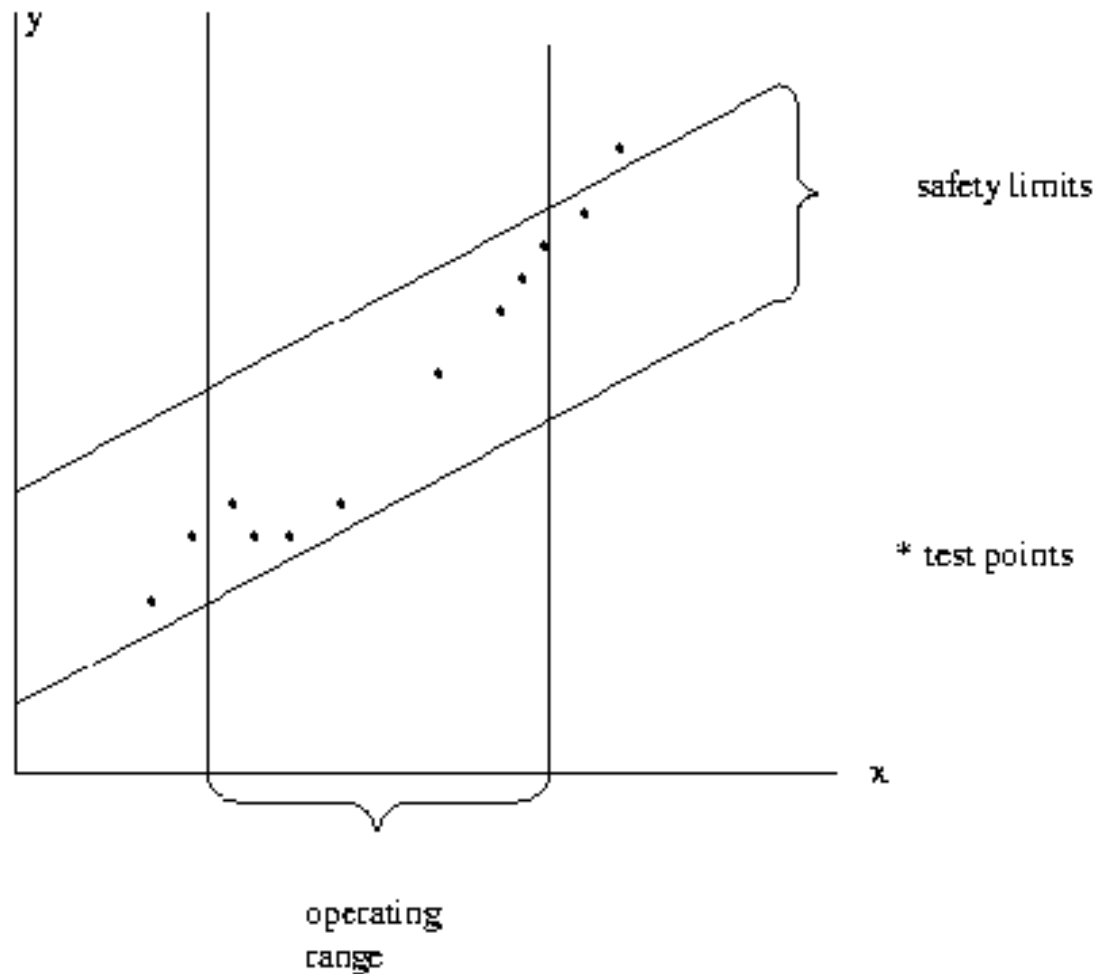


El 23 de abril 1990, Su Majestad la Reina reconoció el éxito del proyecto con el Premio de Su Majestad por Realización Tecnológica al trabajo conjunto de *Inmos Company* y de *Computing Laboratory* de la *Oxford University*.

(C.A.R.Hoare, *The logic of engineering design*)

Errores de programación famosos tristemente (19/20)

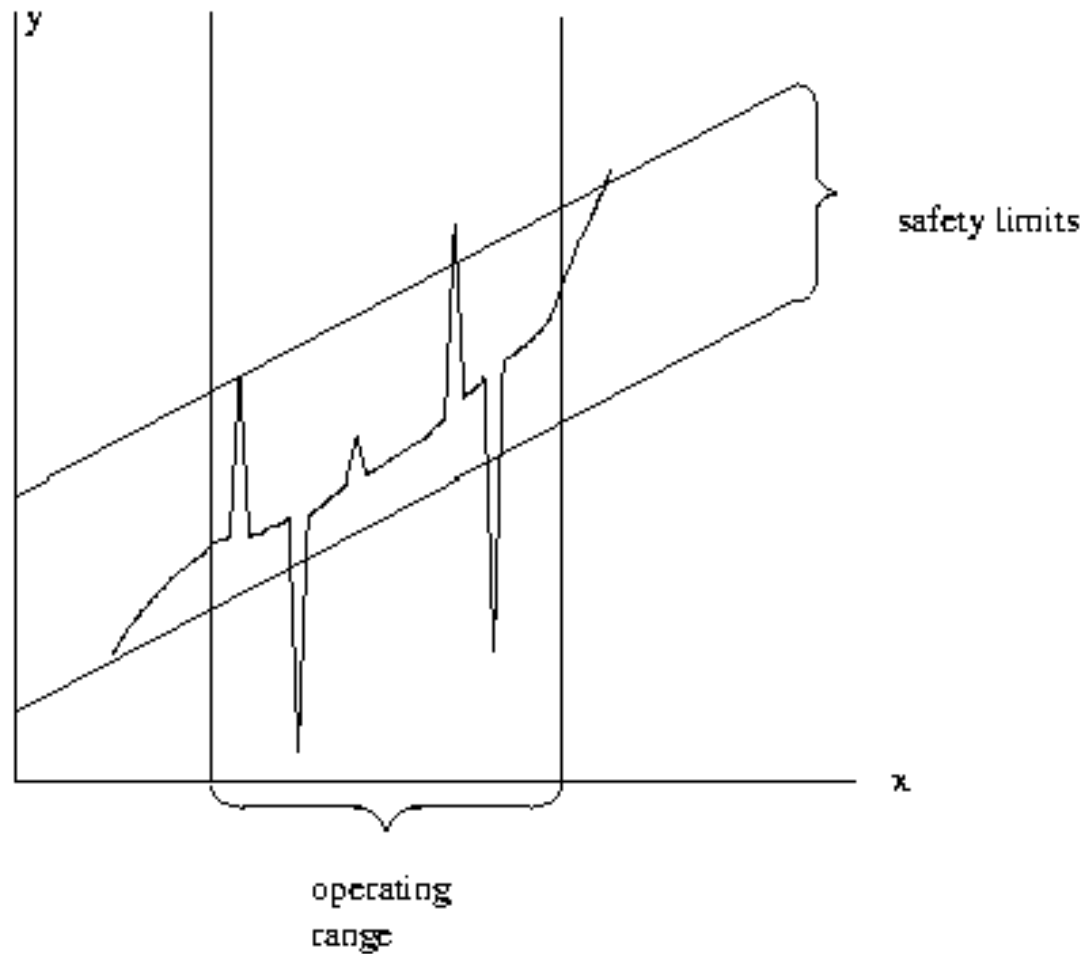
La curva de respuestas en ingeniería



C.A.R. Hoare, *The logic of engineering design*

Errores de programación famosos tristemente (20/20)

La curva de respuestas en programas de computadoras



C.A.R. Hoare, *The logic of engineering design*