Desarrollo del hardware

La aparición de componentes que cada dos años doblan la capacidad de sus antecesores[1] nos ha rodeado en menos de cuatro décadas de máquinas capaces de procesar miles de millones de operaciones por segundo (MTOPS).

En 1946 ENIAC ocupaba una superficie de 160 m2, pesaba 30 toneladas, y ofrecía una capacidad de proceso de 30.000 instrucciones por segundo. En 2002 El microprocesador Pentium IV a 2 Ghz ocupa una superficie de 217 mm2 y tiene una capacidad de proceso de 5.300 MTOPS ("Millions of theoretical operations per second)

En la actualidad son cuatro los factores que imprimen un ritmo acelerado a la industria del hardware.

De ellos, tres son consecuencia de la ley de Moore: Incremento constante de la capacidad de operación, miniaturización y reducción de costes para la producción de hardware; y a éstos se ha sumado en la última década el avance de las comunicaciones entre sistemas. La consecuencia es obvia: ordenadores potentes, que pueden llevarse en el bolsillo y en permanente conexión con grandes sistemas, redes de comunicación públicas, sistemas de localización GPS, etc.

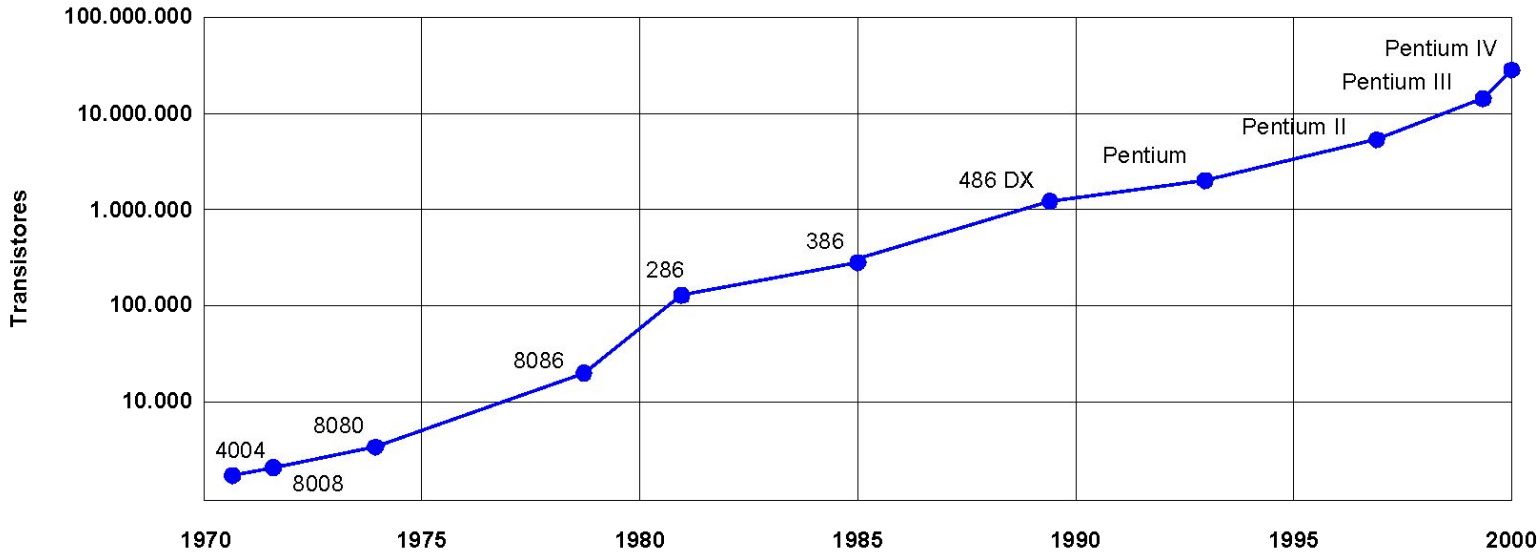
Estas cuatro líneas de avance han extendido el ámbito de aplicación del hardware, e incrementado al mismo ritmo exponencial la complejidad de los sistemas en los que se integra. Los ordenadores ya no son máquinas útiles sólo para la banca o el ejército. Se encuentran presentes en todos los ámbitos, por su capacidad de proceso y de comunicación pueden ofrecer soluciones a sistemas cada vez más complejos.

Este es el escenario creado por la industria del hardware, y que en las tres últimas décadas ha implicado a los desarrolladores de software en retos a los que no han sabido responder con solvencia.

[1] Ley de Moore

Desarrollo del hardware

Desde 1965 la Ley de Moore rige la evolución de los microprocesadores



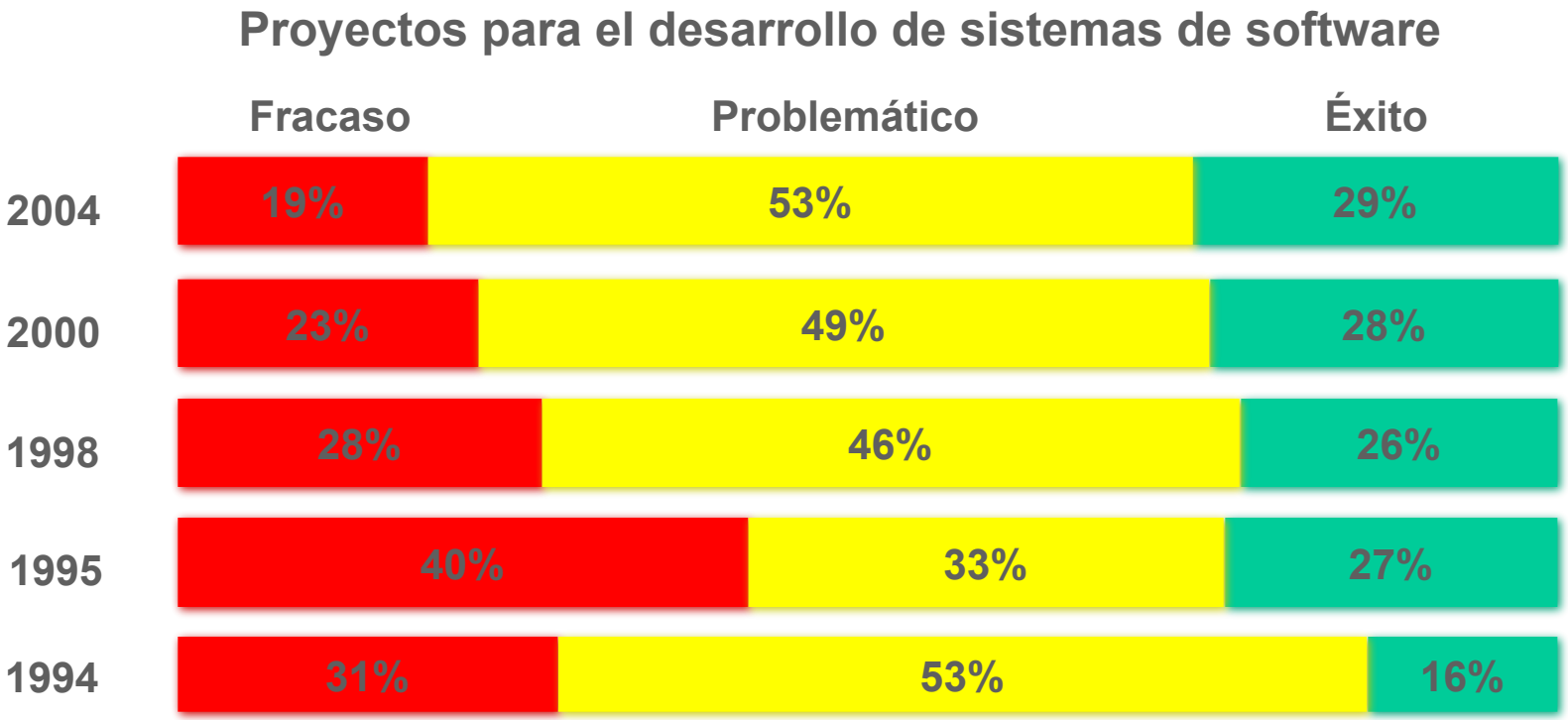
Factores que imprimen aceleración al ritmo de crecimiento del hardware:

Consecuencias de la ley de Moore

- Incremento de la capacidad de operación.
- Incremento de la miniaturización.
- Reducción de costes en la producción.

Comunicaciones entre sistemas

Crisis de software



El proyecto se aborta o el sistema no se llega a utilizar

Desbordamiento de agendas o costes. Las funcionalidades no cubren las expectativas. Problemas funcionales

Proyecto realizado en el tiempo previsto, con los costes previstos, con la funcionalidad esperada y ofreciendo un funcionamiento correcto.

Fuente: Standish Group Survey,

Crisis del software

Este problema se identificó por primera vez en 1968, año en el que la organización NATO desarrolló la primera conferencia sobre desarrollo de software, y en la que se acuñaron los términos "**crisis del software**" para definir a los problemas que surgían en el desarrollo de sistemas de software, e "**ingeniería del software**" para describir el conjunto de conocimientos que existían en aquel estado inicial.

Algunas referencias útiles para comprender cuáles eran los conocimientos estables para el desarrollo de software en 1968 son:

- En 1962 se publicó el primer algoritmo para búsquedas binarias.
- C. Böhm y G. Jacopini publicaron en 1966 el documento que creaba una fundación para la eliminación de "GoTo" y la creación de la programación estructurada.
- En 1968 los programadores se debatían entre el uso de la sentencia GoTo, y la nueva idea de programación estructurada; ese era el caldo de cultivo en el que Edsger Dijkstra escribió su famosa carta "GoTo Statement Considered Harmful" en 1968.
- La primera publicación sobre programación estructurada no vio la luz hasta 1974, publicada por Larry Constantine, Glenford Myers y Wayne Stevens.
- El primer libro sobre métrica de software fue publicado en 1977 por Tom Gilb.
- El primero sobre análisis de requisitos apareció en 1979

Ingeniería del software

Definición original:

> "**Establecimiento y uso de principios de ingeniería para obtener software económico que trabaje de forma eficiente en máquinas reales**".

Fritz Baver, 1968 (conferencia NATO)

Otras definiciones

> "**Disciplina para producir software de calidad desarrollado sobre las agendas y costes previstos y satisfaciendo los requisitos**".

S. Schach 1990, Software Engineering

> "**(1) La aplicación de métodos sistemáticos, disciplinados y cuantificables para el desarrollo, operación y mantenimiento de software; esto es, la aplicación de la ingeniería al software**.
> **(2) El estudio de (1)**".

IEEE 1993

Ingeniería del software

Desde 1968 hasta la fecha han sido muchos los esfuerzos realizados por los departamentos de informática de las universidades, y por organismos de estandarización (SEI, IEEE, ISO) para identificar las causas del problema y definir pautas estándar para la producción y mantenimiento del software.

Los esfuerzos se han encaminado en tres direcciones principales.

- Identificación de los factores clave que determinan la calidad del software.
- Identificación de los procesos necesarios para producir y mantener software.
- Acotación, estructuración y desarrollo de la base de conocimiento necesaria para la producción y mantenimiento de software.

El resultado ha sido la **necesidad de profesionalizar el desarrollo, mantenimiento y operación de los sistemas de software**, introduciendo métodos y formas de trabajo sistemáticos, disciplinados y cuantificables.

La forma de trabajo de programadores individuales surgida por la necesidad de los primeros programas, ha creado una cultura de la programación heroica, para el desarrollo de software que es la principal causa de los problemas apuntados, y en la actualidad una de las principales resistencias a la implantación de técnicas de ingeniería para el desarrollo de sistemas

Estándares y modelos

La Ingeniería del Software es una ingeniería muy joven que necesitaba:

- **Definirse a sí misma**: ¿Cuáles son las áreas de conocimiento que la comprenden?
- **Definir los procesos que intervienen en el desarrollo, mantenimiento y operación del software**
- **De las mejores prácticas, extraer modelos de cómo ejecutar esos procesos para evitar los problemas de la "crisis del software"**
- **Definir criterios unificadores para las tareas de requisitos, pruebas, gestión de la configuración, etc.**

Los estándares son útiles porque:

- Agrupan lo mejor y más apropiado de las buenas prácticas y usos del desarrollo de software.
- Engloban los "conocimientos".
- Proporcionan un marco para implementar procedimientos de aseguramiento de la calidad.
- Proporcionan continuidad y entendimiento entre el trabajo de personas y organizaciones distintas.