Introducción a los métodos formales

Adrián Riesco

Universidad Complutense de Madrid, Madrid, Spain

Especificación, Validación y Testing 2014/15

Introducción a los métodos formales

1. Introducción
2. Los métodos formales
3. Ingeniería del software automática

Motivación

- Nuestra sociedad depende de la calidad del software:
  - Fiabilidad.
  - Seguridad.
  - Robustez.
  - Corrección.
  - Eficiencia.
- . . . incluso en el software de consumo.

Motivación

Hace 15 años. Las garantías de calidad eran necesarias en áreas de seguridad crítica.

Hoy en día. Las consecuencias de los fallos software son inaceptables en todos los ámbitos.

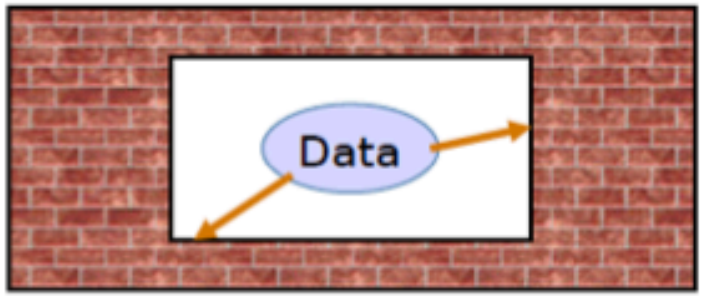
- Se buscan sistemas que nunca deben fallar y siempre deben cumplir los plazos de entrega.
- Deben ser disponibles, confiables, fiables, mantenibles y seguros.
- Existen riesgos en los sistemas de rendimiento crítico en sistemas abiertos e interconectados que producen grandes pérdidas si se cuelgan

Fiabilidad frente a seguridad

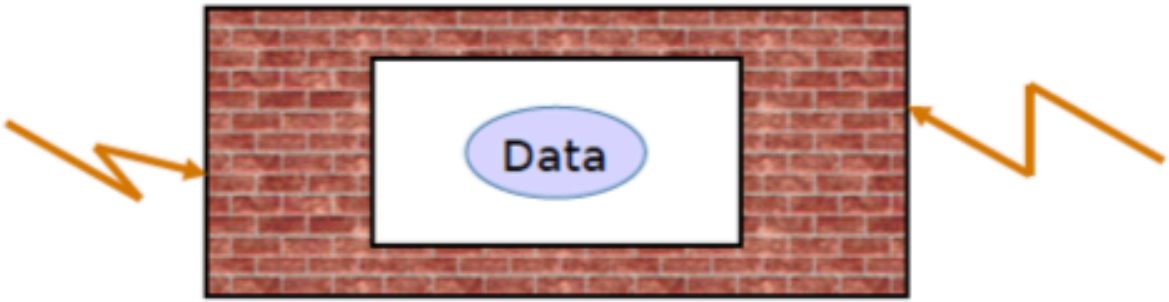
- La fiabilidad se refiere a propiedades:
  - que aseguran que el entorno está protegido del sistema; y
  - y que se formulan en términos de que ciertos eventos no suceden (e.g., "La temperatura del reactor es menor que 500° C").
- La seguridad es una "hiperpropiedad":
  - Se asegura que el sistema está protegido del entorno (comportamiento correcto ante un adversario inteligente).
  - Diferentes tipos de propiedades de seguridad:
    - Confidencialidad o "no hay revelación improcedente de información", como secreto de claves, anonimato de la identidad, autentificación, privacidad de datos (e.g. información de tarjetas de crédito), etc.
    - Integridad o no modificación improcedente de información, como falsificación (e.g. "el mensaje enviado es el que llegará"), repudiación, corrupción, etc.

Fiabilidad frente a seguridad

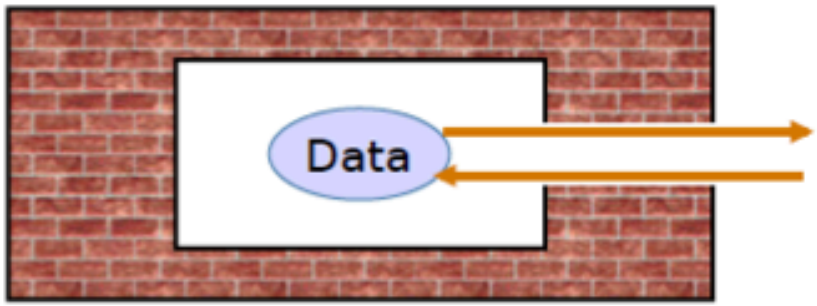
- Confidencialidad



- Integridad



- Disponibilidad



Historias de errores

home > philosophy > motivation

Why is Formal Methods Necessary?

Digital systems can fail in catastrophic ways leading to death or tremendous financial loss. Although their are many potential causes including physical failure, human error, and environmental factors, **design errors** are increasingly becoming the most serious culprit. The following recent events show the potential of design errors for disaster (Note: the external links, which are labeled by , should not be construed to represent endorsement of any commercial products):

- The maiden flight of the Ariane 5 launcher (June 4 1996) ended in an explosion. Total loss was over $850 million.
- Between June 1985 and January 1987, a computer-controlled radiation therapy machine, called the Therac-25 , , massively overdosed six people, killing two.
- Replacement of defective Pentium processors costs Intel Corp. several hundreds of millions of dollars in 1995.

Historias de errores

- Replacement of defective Pentium processors costs Intel Corp. several hundreds of millions of dollars in 1995.
- The April 30, 1999 loss of a Titan I , which cost the taxpayers $1.23-billion, was due to incorrect software (incorrectly entered roll rate filter constant)
- December 1999 loss of the Mars Polar Lander was due to an incomplete software requirement. A landing leg jolt caused engine shutdown.
- Denver Airport's computerized baggage handling system delayed opening by 16 months. Airport cost was $3.2 billion over budget.
- Patriot failure at Dharan (software error put tracking off by 0.34 of a second)
- NASA's Checkout Launch and Control System (CLCS) cancelled 9/2002 after spending over $300 million.
- BYTE Magazine's Notorious Bugs"

`http://shemesh.larc.nasa.gov/fm/fm-why-new.html`

Historias de errores

Un error informático vuelve a cuestionar la seguridad de Ascó

Un incidente afectó al ordenador que alerta de la presencia de gases tóxicos

Por: JORDI SIRÉ
14/10/2008 21:10:00

La incertidumbre sobre la seguridad nuclear recayó el lunes en la planta número uno de Ascó. Un incidente afectó al ordenador que alerta de la presencia de gases tóxicos. "No es un ordenador de administración, sino el que controla parte del proceso de generación", admitieron fuentes sindicales, poco dadas a sobredimensionar los achaques regulares de las plantas por temor a difuminar los incidentes realmente importantes.
Según la nota hecha pública por el Consejo de Seguridad Nuclear (CSN), en el incidente fallaron dos de los trenes de detección de posibles gases nocivos en el centro neurálgico de la central, contagiando el error al ordenador y obligando a poner en funcionamiento el sistema de ventilación de emergencia de la sala como medida preventiva.
El CSN calificó el suceso como nivel 0 en la escala internacional porque no comportó riesgo para las personas o el medioambiente. El portavoz de la campaña de energía nuclear de Greenpeace, Carlos Bravo, lamentó el repetido fallo de los aparatos de medición de Ascó I y II: "En estos casos emiten señales falsas y cabe tener en cuenta que las decisiones se toman a partir de los datos que proporcionan estos ordenadores", declaró, comparando el incidente con un fallo en el cuentakilómetros de un coche.

`http://m.publico.es/164793`

Historias de errores

Un error informático en la T-4 de Barajas provocó la pérdida de 20.000 maletas

Posteriormente se averió un tren lanzadera que permite a los pasajeros desplazarse por la terminal

3 de abril de 2006

La nueva terminal T-4 del aeropuerto de Barajas sigue dando de qué hablar. Unas 20.000 maletas se perdieron el sábado debido a un error informático que truncó la clasificación de equipajes. Este fallo colapsó durante cinco horas las cintas encargadas de trasladar y clasificar los bultos. Estos no pudieron embarcar en sus vuelos correspondientes -que en muchos casos despegaron sin apenas maletas- y se quedaron acumulados en el interior de la terminal, según informaron empleados del aeropuerto.

Siemens, la empresa encargada del diseño y mantenimiento del dispositivo, considera excesiva la cifra ofrecida por los empleados, sobre todo teniendo en cuenta que la facturación actual de todo un día en la nueva terminal puede oscilar precisamente entre las 20.000 maletas citadas por los trabajadores y los 30.000 bultos.

Siemens aseguró que, tras la incidencia del sábado por la mañana, el domingo el sistema funcionaba "con absoluta normalidad" y que los equipajes facturados desde entonces "no han sufrido ningún problema".

Iberia, la compañía aérea que opera en la T-4, empezó el mismo sábado por la tarde a embarcar maletas en aviones para que los pasajeros que se quedaron con lo puesto pudieran disponer de sus enseres cuanto antes. Incluso ha decidido utilizar camiones para distribuir por toda España aquellos bultos que debían haberse cargado en vuelos nacionales. "No sabemos cuántos hay que mandar al extranjero ni cuándo terminaremos, pero posiblemente mañana -por hoy- estará enviado todo el equipaje", señaló un portavoz de Iberia.

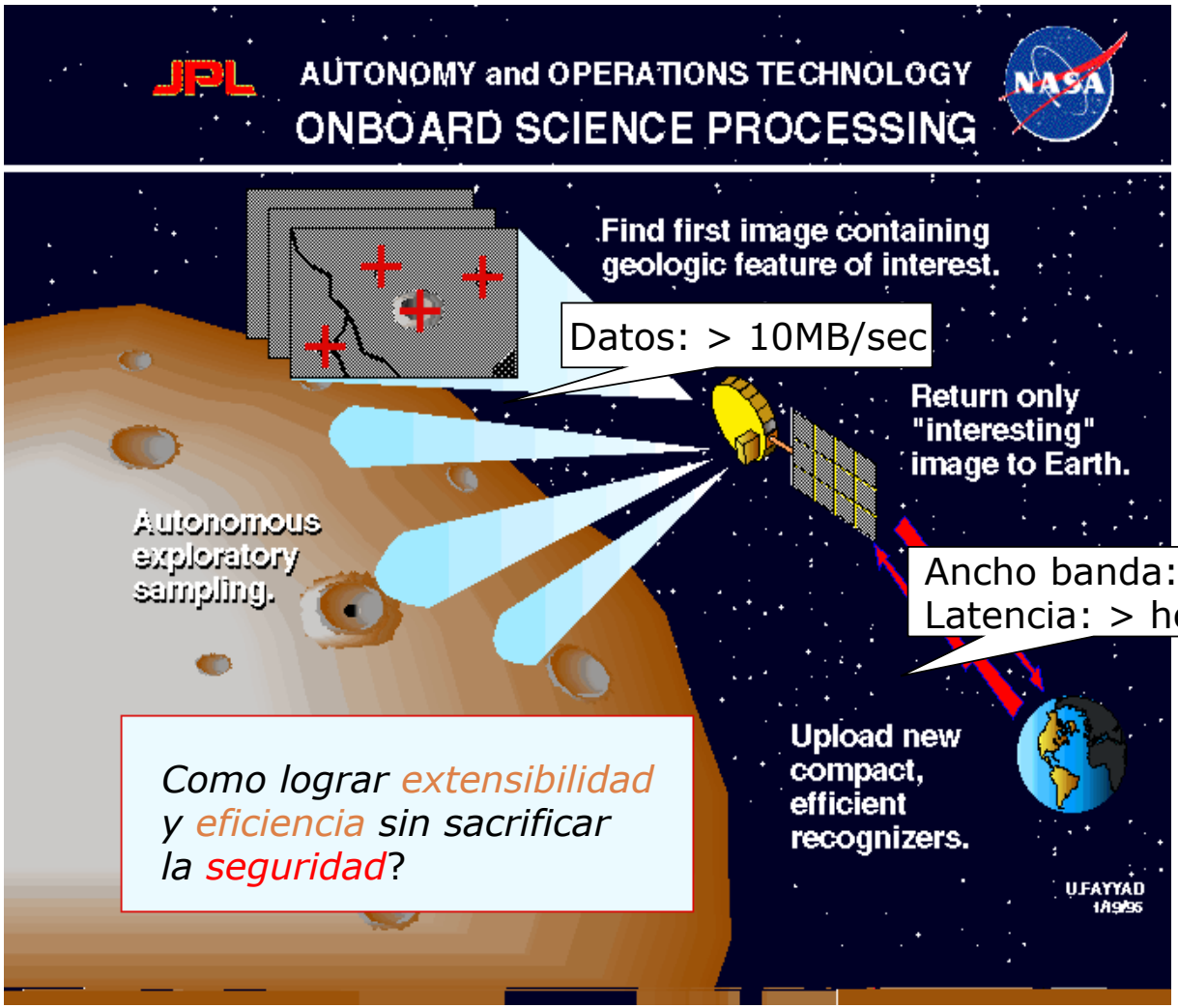
La crisis de la Ingeniería del Software

- La distinción entre "áreas de seguridad crítica" e "informática de consumo" se va reduciendo.
- Los fallos del software implican catástrofes con graves pérdidas económicas.
- Estas pérdidas ascienden a un 9 %, ocupando el 5° puesto, detrás de los incendios, el terrorismo, los huracanes y los terremotos.
- Alguien deberá proporcionar tecnología para garantizar la calidad del software.

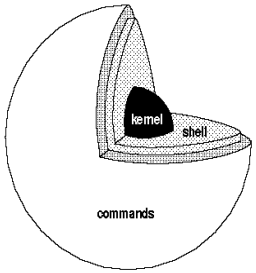
Hechos constatados en Ingeniera del Software clásica

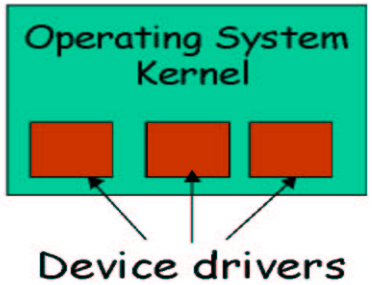
- Los errores son caros.
- El mantenimiento de un producto software es 2/3 del coste total.
- Solventar los errores en la fase de mantenimiento es hasta 20 veces más caro que en la fase de diseño.
- Los errores dañan el prestigio de los fabricantes de software y la confianza en sus productos:
  - Castigo de los consumidores a INTEL por el error del Pentium
  - A Chrysler por los airbag laterales (programados en dirección contraria al impacto).
  - Al Eurofighter (reducción de ventas previstas).
  - Sistemas donde el error es inaceptable (pérdidas humanas, caso del Therac-25).

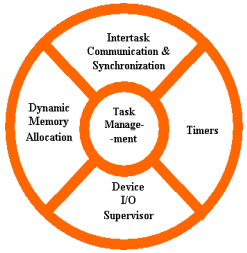
Software de alta seguridad



Software de consumo

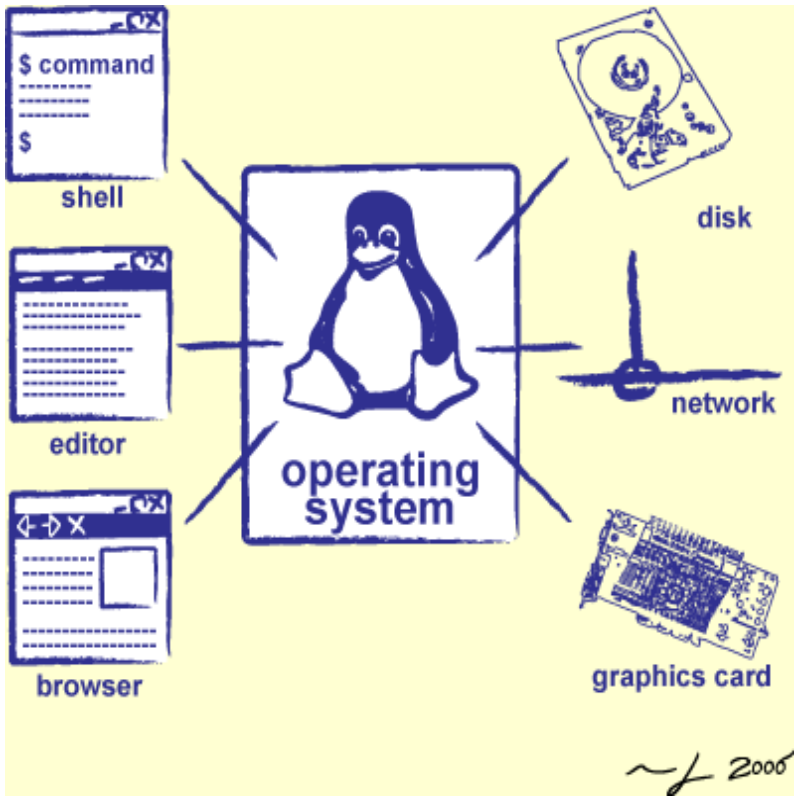






- Podemos encontrar ejemplos en la vida diaria.
- Los drivers son necesarios para la correcta ejecución de un SO.
- Proceden de fuentes remotas posiblemente no veraces.
- Se ejecutan en el espacio de direcciones del núcleo del SO.
- Deberían poderse comprobar antes de ejecutarlos.

Software de alta seguridad



- El código móvil necesita garantías especiales de calidad.

Software de consumo

- El número de teléfonos móviles ha sobrepasado el de ordenadores (7000M teléfonos móviles vs 1500M PCs; tantos móviles como personas en el planeta).
- La gran mayoría disponen de conexión inalámbrica.
- La eficiencia energética y el ancho de banda son las características más demandadas.
- Sin embargo, la seguridad no es algo que suela tenerse en cuenta.

Aprendiendo de los errores

- Analicemos las causas más frecuentes de fallo en un sistema software que ha sido rigurosamente diseñado y probado exhaustivamente.
- Y después veremos que "defectos" muy similares a los que son responsables de los errores no intencionados son los mismos que se usan al diseñar software intencionalmente malicioso.

Ariane 5 (ESA)

- El 4 de Junio de 1996 el Ariane 5 despeg como estaba previsto.
- 40 segundos más tarde explota.
- Más tarde se descubrió que fue debido a un fallo al reutilizar software del Ariane 4 (números reales de 64 bits convertidos a enteros de 16 bits).
- Durante los siguientes años, muchos libros y presentaciones de Ingeniería del Software han usado esta idea.

Ariane 5 (ESA)

- Este accidente tuvo unas consecuencias de unos 7.000 millones de euros.
- Hasta el Ariane, no se apreciaba cómo el software puede contribuir al fallo de los sistemas.
- "El software no puede fallar" era una creencia popular.
  - El programa Ariane 5 estuvo en peligro.
  - También el SOHO (Estudio de la heliosfera del Sol).
  - Muchos investigadores perdieron su puesto.
  - El futuro de los satélites dejó de estar asegurado.
- Las técnicas de replicación de componentes, usadas para mejorar la tolerancia a fallos en el hardware, no sirvieron para prevenir este error.

Problemas en la NASA

- En septiembre de 1999, la NASA pierde el Mars Polar Lander y el Mars surveyor Climate Orbiter.
- Se determinó que los errores se deban a:
  - En el caso del Lander, a una violación de precondición (kilómetros en lugar de millas).
  - En el caso del Orbiter, a la reutilización de un componente: un subcontratista usaba unidades britnicas (libra-segundo) en vez de unidades decimales como la NASA (newton-segundo).

Problemas en la marina de EEUU

- El crucero de guerra Yorktown también sufrió un error informático.
- Después de que un miembro de la tripulación introdujera un cero en un campo de la aplicación, el ordenador procedió a hacer una división por cero.
- La operación causó que el sistema de propulsión se parase.
- Resultado: el Yorktown estuvo a la deriva durante más de dos días.

Conclusiones a estos problemas

- Un agente malintencionado puede atacar la seguridad.
- Para ello se debe aprovechar de las vulnerabilidades:
  - Fallos en la validación de campos de entrada.
  - Desbordamiento de memoria (e.g., división entre 0).
  - VBS (Visual Basic Scripts).

Garantías

- ¿Qué garantías se pueden proporcionar?
- **Garantía limitada:** Microsoft garantiza que el producto de software funcionará en concordancia con la documentación que lo acompaña durante un período de 90 días desde la fecha de recepción.
- **Ninguna responsabilidad:** En ningún caso Microsoft o sus proveedores serán responsables por cualquier daño ESPECIAL, ACCIDENTAL, INDIRECTO, O RESULTANTE derivado del uso del producto software... la responsabilidad de Microsoft se limita al máximo entre la cantidad pagada por el producto software y 10 dólares.

Solución

- La universidad Carnegie Mellon encontró una solución para que la Ingeniería del Software sea capaz de responder a la calidad que se le supone.
- Poder garantizar altos niveles de confianza depende fundamentalmente de nuestra habilidad para razonar sobre los programas.
- Para ello usaremos los Métodos Formales.
- Es una excelente oportunidad de aplicación para:
  - La lógica computacional.
  - La Teoría de tipos.
  - Las Semánticas formales.

Métodos formales

- En general hay dos corrientes "extremas" opuestas.
- Quienes opinan que "todo lo importante se puede demostrar formalmente".
- Quienes opinan que "nada demostrable es importante".
- El término medio suele ser el más realista.
- Las especificaciones, los modelos o las semánticas están cobrando una importancia creciente en gran diversidad de productos software comerciales (incluyendo herramientas para Java o UML).

Métodos formales

- Programas Marco y Horizonte 2020 de la Unión Europea: "La calidad del software es fundamental para el liderazgo europeo en la economía del conocimiento".
- EEUU — Informe PITAC: "La economía y seguridad americanas dependen del software y este se comporta de manera peligrosa. Las necesidades de corrección crecen más deprisa que la capacidad actual de producir software demostradamente correcto".
- Por ello se recomienda "aumentar la inversión en Métodos Formales para desarrollar la tecnología que demanda la Ingeniería del Software del siglo XXI".

Las partes de la solución

- Las soluciones más satisfactorias que se plantean están basadas en la integración y el uso combinado de:
  - Testing y simulación.
  - Métodos formales.

Testing

- El principal problema del testing está en proporcionar un conjunto de datos de entrada representativo y en comparar la salida con el resultado esperado.
- Por un lado, los tests exhaustivos son imposibles. . . (normalmente se trabaja un dominio infinito de datos, como los números enteros o los reales).
- Y respecto a los tests representativos. . . los problemas suelen ocurrir cuando se proporcionan como entrada datos inesperados o inusuales.

Demostraciones vs. experimentos

- En principio, la corrección del hardware o de un programa debería probarse como un teorema matemático.
- De hecho, en Matemáticas, lo normal es demostrar los resultados rigurosamente (es imposible imaginar a un matemático haciendo testing para demostrar un teorema).
- Sin embargo, en Informática, lo normal es establecer las garantías de funcionamiento mediante simple testing empírico (aquí lo "imposible" de imaginar es a un informático demostrando teoremas sobre sus programas).

La importancia de las demostraciones

- La función $\pi(n) - \int_0^n du/ln(u)$ mantiene su signo para $n \leq 10^{10}$.
- También el algoritmo de división del Pentium funcionó perfectamente en billones de casos de prueba y aplicaciones reales.
- Sin embargo, en ambos casos la conclusión natural (inductiva) fue 100 % equivocada.
- Conclusión: El testing puede pasar por alto escenarios y detalles relevantes que solo se pondrán de manifiesto en una demostración formal.

La importancia de la verificación formal

- La verificación formal aporta beneficios muy importantes:
  - Permite encontrar en el software errores aún no descubiertos.
  - Puede revelar formas de mejorar la eficiencia de los algoritmos.
  - Mejora nuestra confianza en dichos algoritmos.
  - Mejora la comprensión global del sistema.

Los métodos formales

1. Introducción
2. Los métodos formales
3. Ingeniería del software automática

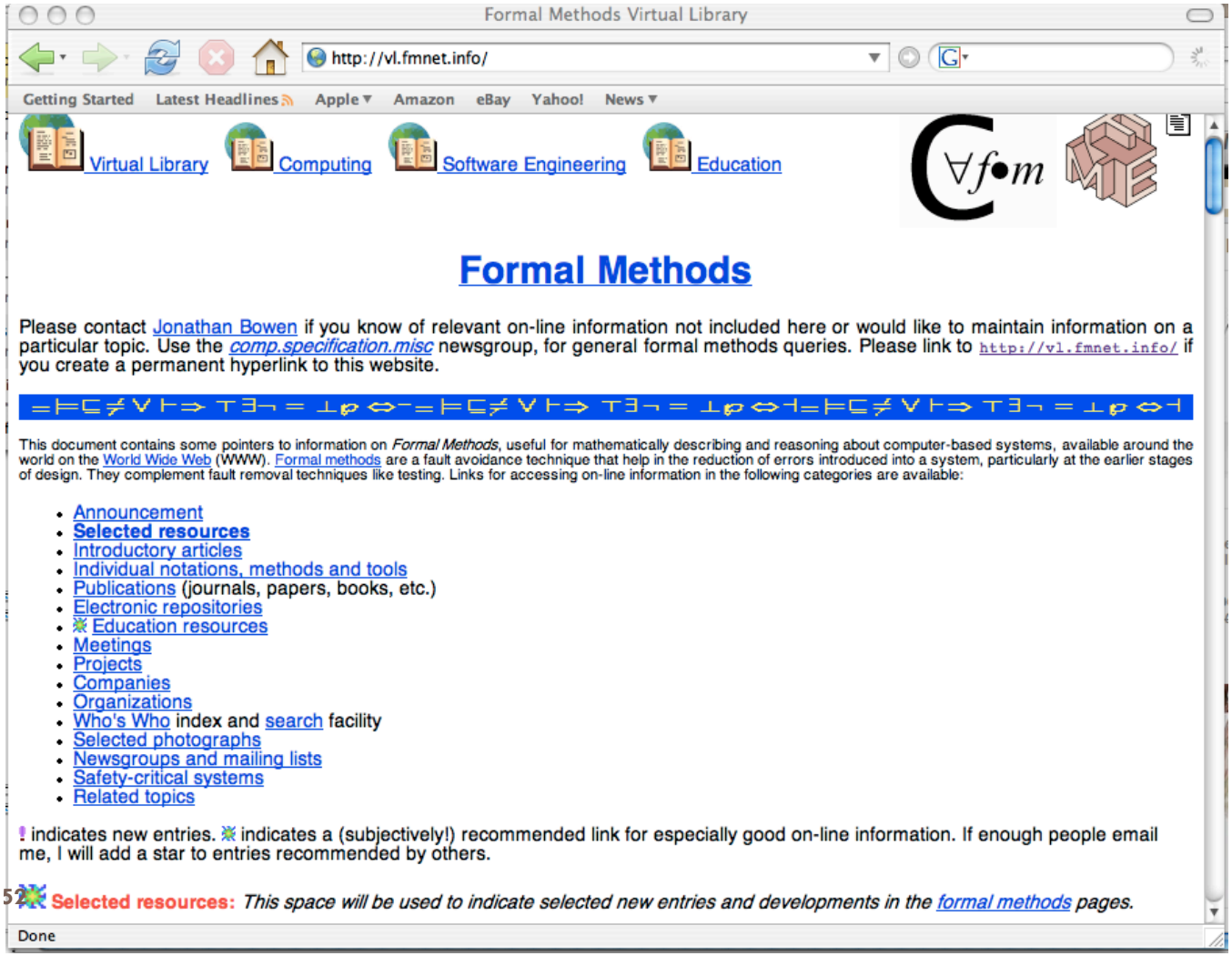
¿Qué son los métodos formales?

- Según J. Wordsworth, es "Un método formal (de desarrollo SW) es un proceso (para desarrollar SW) que aprovecha la potencia de la notación y pruebas matemáticas".
- Según D. Schmidt, es "Un método formal fuerte es aquél soportado por una herramienta automática".

Ventajas de los métodos formales

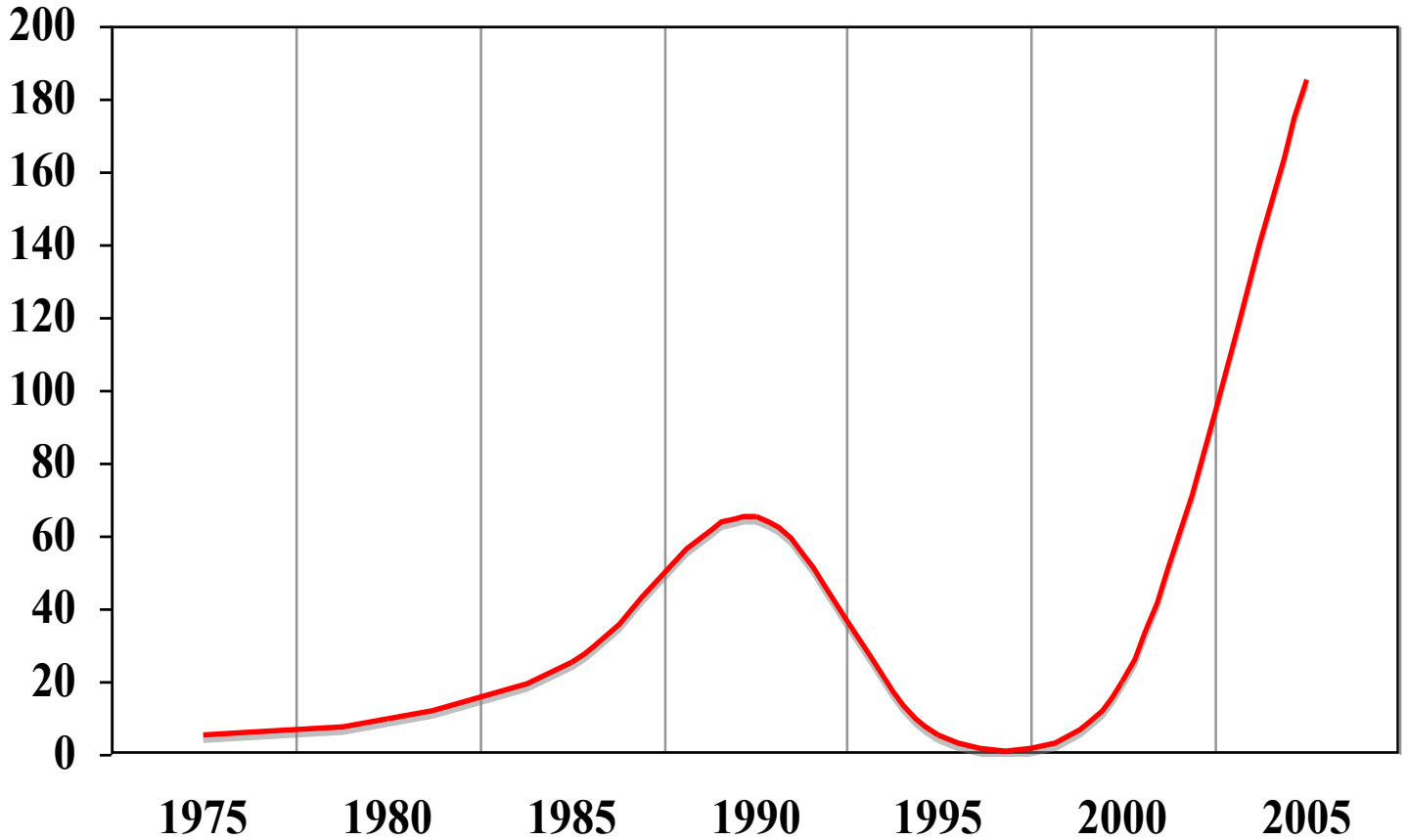
- Menores costes de desarrollo y (sobre todo) mantenimiento.
  - Detección precoz de errores, verificación más simple, mayor fiabilidad...
- Mejor comprensión del sistema
  - Clarificación de requerimientos de usuario, buena documentación...

Evolución de los métodos formales



http://formalmethods.wikia.com/wiki/VL

Evolución de los métodos formales



Heavy and weak FMs
(Pesados y débiles)

Lightweight and strong FMs
(Ligeros/Ágiles y fuertes)

Ingeniería del software automática

1. Introducción
2. Los métodos formales
3. Ingeniería del software automática

Ingeniería del software automática

- Algunas herramientas de métodos formales parece que requieran una tesis para poder usarlas.
- Aunque no sea el caso, la etiqueta formal "asusta" a muchos ingenieros de software.
- Es necesario un cambio en la manera de pensar, tanto por parte de usuarios como de los programadores de las herramientas.
- Por ello, muchas de las herramientas formales en la actualidad son automáticas.

Aproximación ligera/ágil

- A diferencia de otros enfoques formales más convencionales, que fomentan la formalización excesiva mediante el empleo de lenguajes demasiado expresivos y que requieren una formacin matemtica poco habitual en los usuarios finales.
- La aproximación ligera, basada en la aplicación selectiva y focalizada de los métodos formales, resulta más efectiva y rentable en la práctica.

Características de la aproximación ligera

Parcialidad como criterio (en la expresividad, en la especificación, etc).

Selectividad. Es decir, adaptación al dominio.

Integración. Combinación con métodos convencionales.

Transparencia del formalismo al usuario.

Bajo coste. En general con tecnología declarativa.

Éxitos de los métodos formales: Météor

- El primero éxito real fue la línea 14 del metro de París llamada Météor (MÉTro Est-Ouest Rapide) que funciona sin conductor.
- Se escribieron unas 110.000 líneas de modelos en B, generando unas 86.000 líneas de código Ada.
- Participaron las empresas Alstom y Siemens.
- No se ha detectado ni un solo error después de las pruebas del código: ni en la fase de validación funcional ni en la integración ni en las pruebas in-situ, ni tampoco desde que la línea comenzó a funcionar (octubre 1998).
- El software desarrollado sigue todavía en la versión 1.0, sin ningún error detectado hasta la fecha.
- En 2007 se extendió la línea, en marzo de 2014 se extendió de nuevo y se planean más extensiones para 2015.

`http://en.wikipedia.org/wiki/Paris_M%C3%A9tro_Line_14`

Éxitos de los métodos formales: Astrée

- El desarrollo del sistema Astrée se inició en noviembre de 2001.
- Principales aplicaciones: análisis estático de programas escritos en C para sistemas síncronos, basados en eventos, de tiempo real, fiabilidad crítica, software embebido, etc.
- Éxitos consegidos:

  Noviembre 2003. Astrée fue capaz de probar automáticamente la ausencia de errores en el software de control de vuelo del Airbus A340, un programa de 132.000 líneas de C analizado en 1h20 en un PC 32-bit 2.8 GHz usando 300 Mb de memoria.

  Enero 2004. Se extendió Astrée para analizar el código de control de vuelo de la serie A380 de Airbus France, y se completó a tiempo para el primer vuelo el 27 de abril de 2005.

  Abril 2008. Astrée fue capaz de probar automáticamente la ausencia de errores en una versión en C del sistema de acoplamiento del Jule Vernes Automated Transfer Vehicle (ATV) desarrollado por la ESA para la Estación Espacial Internacional.

Éxitos de los métodos formales: Compilación certificada

- Compcert es un compilador que genera cdigo ensamblador de PowerPC a partir de un amplio subconjunto del lenguaje C (Clight).
- La particularidad de este compilador es que está escrito en el lenguaje del demostrador de teoremas Coq y, por lo tanto, su corrección ha sido automáticamente probada en Coq.