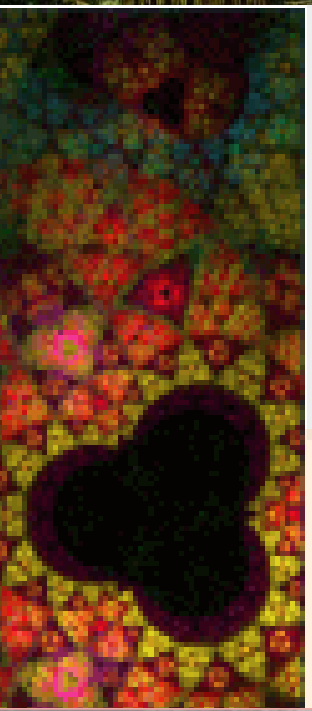
imdea software

instituto madrileño de estudios avanzados



Las Tecnologías de Desarrollo Software: desafíos en la investigación de frontera

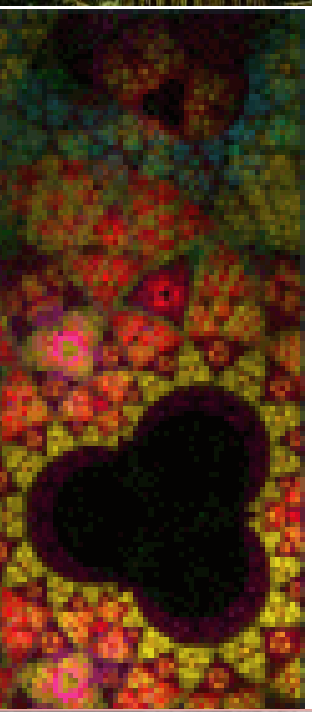
Juan José Moreno-Navarro
IMDEA-Software & UPM
jjmoreno@fi.upm.es

Madrid,

14 Noviembre 2007

imdea software

instituto madrileño de estudios avanzados



Madrid,

14 Noviembre 2007

Programas de computador

- **Un programa de ordenador son un conjunto de instrucciones que le dicen al computador qué tiene que hacer. Esas instrucciones se escriben en un fichero y luego se traducen para que el ordenador las entienda.**
- **Nuestro ordenador no habla idiomas, no entiende español, inglés, francés, alemán o chino. El ordenador entiende 0 y 1:**

  10001000 00000010 00000001

  puede significar ``Suma 2 más 1''.
- **Para pintar un cuadrado azul en la pantalla podría ser necesario escribir más de 100 instrucciones. ¡Imagina escribir a mano 2400 ceros y unos sin cometer ningún error !**

Lenguajes de programación

- **Para no tener que hacer eso, inventamos idiomas nuevos con los que decirle al computador lo que queremos que haga.**
- **Son los *lenguajes de programación*.**
- **Pintar un cuadrado es fácil, pero...**

¡Imagina hacer un juego: millones y millones de cuadrados y triángulos de diferentes colores para que parezcan personas que juegan al fútbol y que hacen lo que les dices!

- **Escribir programas de computador es muy difícil, y es muy común que haya fallos.**

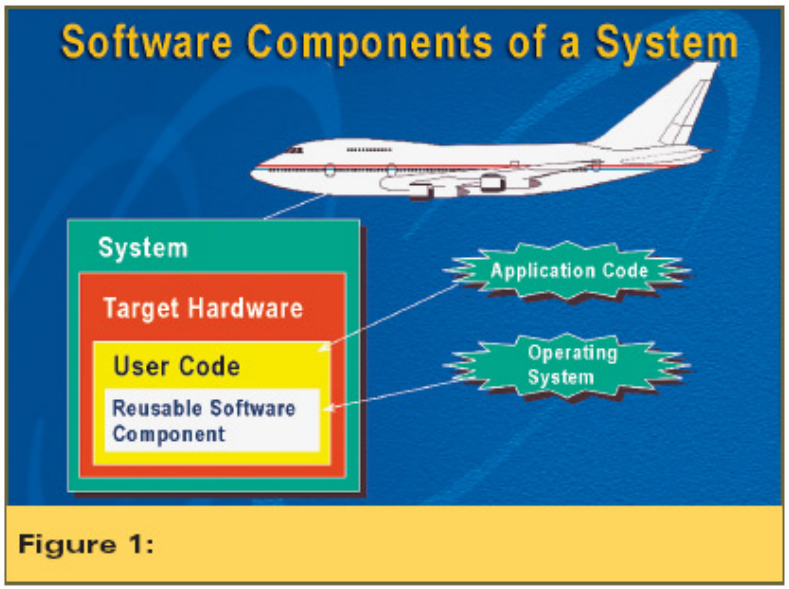
Software por todos lados

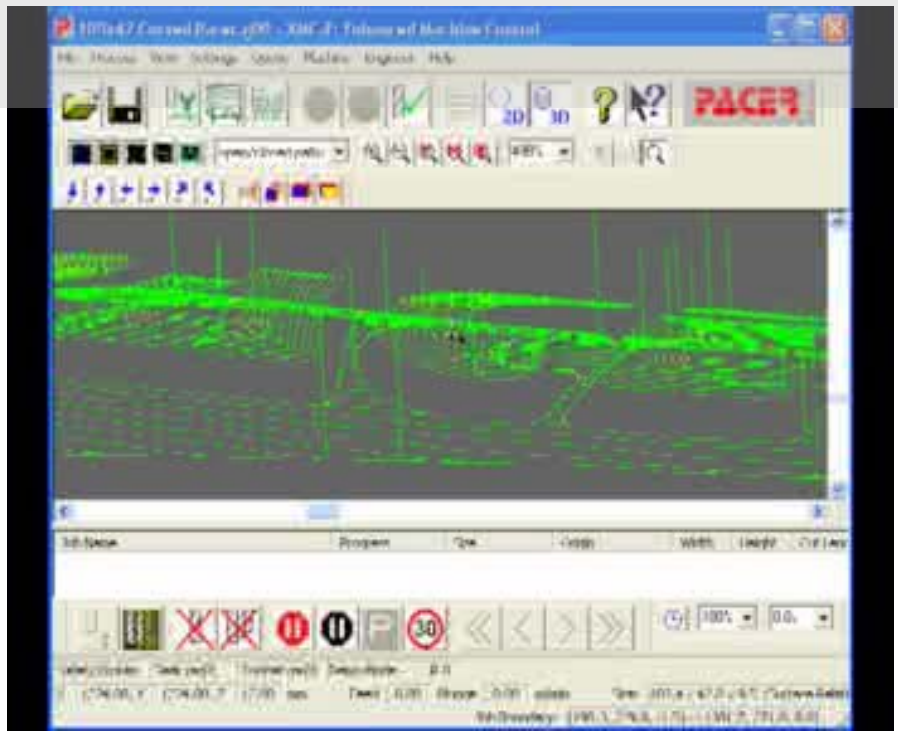
- Generalmente pensamos que sólo hay programas en el ordenador, pero el software está en todas partes en nuestra vida cotidiana.
- Hay máquinas que llevan programas dentro que les dicen qué tienen que hacer: Lavadoras, neveras, vídeos, coches, aviones,...

  Hoy en día, casi todos los aparatos con algo eléctrico llevan un programa.
- Si el programa de una lavadora falla, podría estropearse, o lavar mal. ¿Te imaginas que un avión se cayese porque los programas que lleva fallasen? ¿O que un coche no frenase?

Software en aviones

- Los aviones son máquinas complejas donde una cantidad enormes de funciones están controladas por software.
- Un 45% del gasto de investigación en el diseño de un avión es Software.





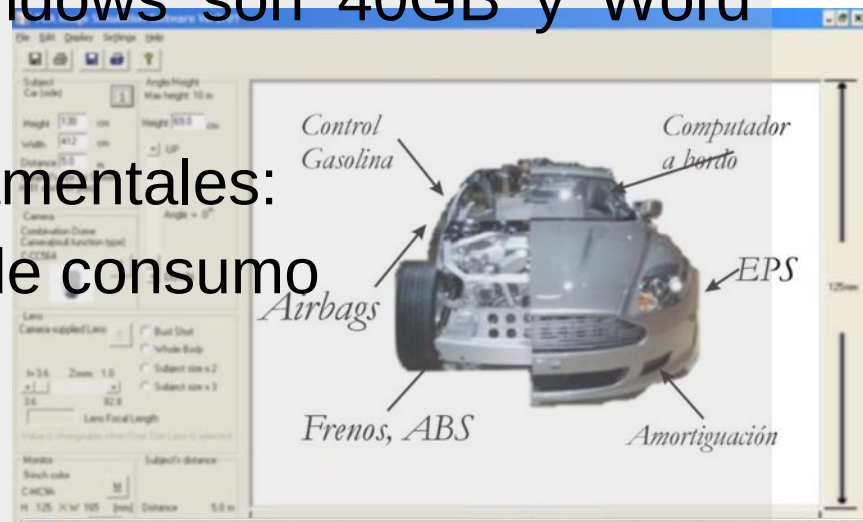
Software en aviones

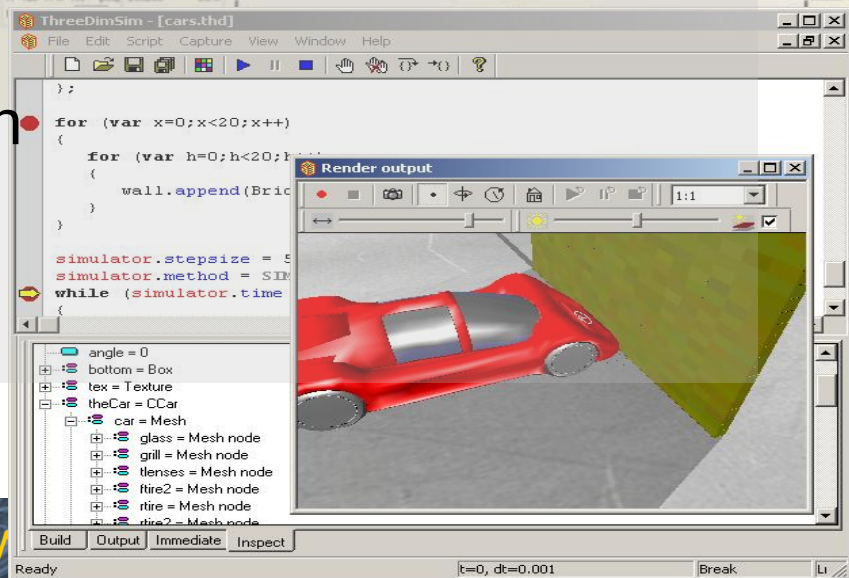
- Por ejemplo, el nuevo AIRBUS A380 necesita 225.000 líneas de código en C.

- Pero debe estar completamente probado, por seguridad y por necesidad. ¿Os imagináis que en pleno vuelo, un avión deba de parar pues tiene que actualizar su SW pues se ha descubierto un error y hay que arreglarlo como pasa con Windows?

Software en automóviles

- Un moderno automóvil lleva instalado más de 4GB de software (por ejemplo, todo Windows son 40GB y Word solo 1,1MB) .
- El Software controla tareas fundamentales:
  - Frenos
  - ABS
  - Control de tracción
  - Control de consumo
  - Airbags
  - EPS
- Un 35% del gasto de investigación en la industria del automóvil es en Software.





Software en otros medios de transporte

- Los trenes de alta velocidad (AVE) pueden ir a velocidades superiores a los 300Km hora. Un conductor humano no puede controlar su funcionamiento y seguridad. Lo hace un programa software.

- Actualmente este Software se prueba con un sistema de prueba y fallo: Debe recorrer X Kilómetros sin presentar ninguna deficiencia:
  - Costoso
  - Poco fiable

Software en otros medios de transporte

- La línea 14 del metro de París se ha hecho famosa por ser la primera línea de metro totalmente controlada por software. !Los coches van sin conductor¡

- El software está completamente verificado con una técnica muy avanzada.

Software en el espacio

- En 2004 la Mars Rover de la NASA se "congeló" en medio de Marte tras un viaje de más de 6 meses y 487 millones de kilómetros. Estuvo tres días parada hasta que se solucionó el problema.
- Phobos 1:
  Perdido en 1988
- ESA Ariane 5, Fligh 501:
  Autodestruido a los 40 minutos del despegue.
- Mars Climate Orbiter:
  Se estrelló al entrar en órbita
- Mars Polar Lander:
  Perdida en 1999

Software en la salud

- Los modernos aparatos de diagnóstico, tratamiento, prevención y terapia funcionan gracias al software que llevan dentro.
- Incidente Panamá: Uso de software no fiable para un dispositivo de tratamiento de cáncer que supuso la muerte de varios pacientes por sobre-exposición a la radiación. Ha llevado a sus autores a ser condenados por los tribunales.
- Actualmente en la investigación de equipos médicos, el software representa el 33%

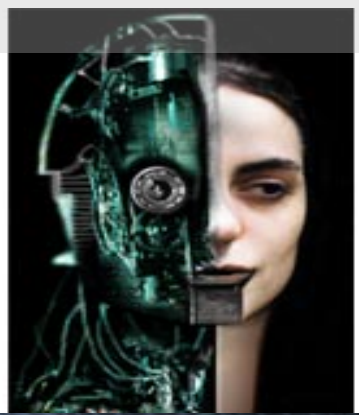
Software en las comunicaciones

- Las comunicaciones no se conciben sin software. La tecnología de un teléfono móvil se conoce desde los 70 (comercial). Todas las mejoras posteriores se basan fuertemente en el software.

- Las industrias de telecomunicaciones gastan hoy un 65% de su investigación en software.

Software en el ocio

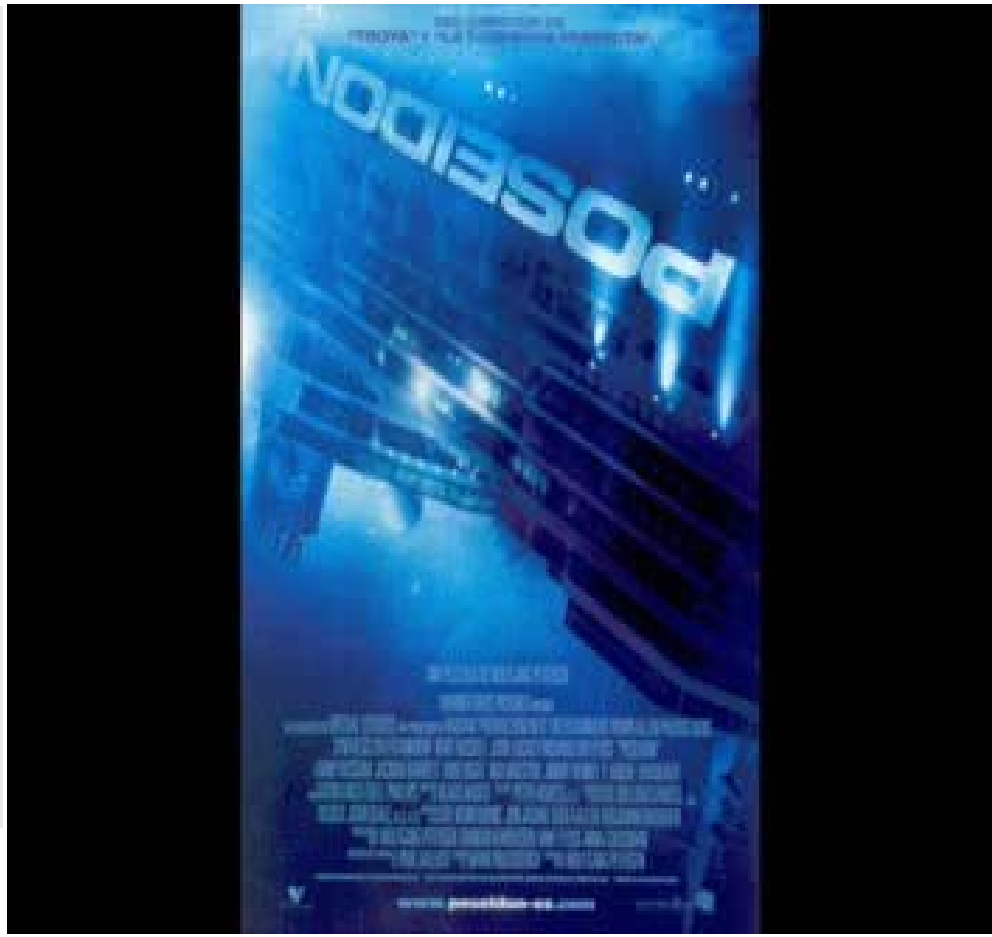
- El software es parte fundamental de la televisión digital, la electrónica de consumo, los videojuegos, las películas, ...
- Este software no necesita fiabilidad absoluta, pero es extremadamente complejo en general y debe responder en un tiempo preciso.
- Necesita buenas herramienta de diseño y a la vez herramientas de optimización.
- Se estima que un total del 60% de los gastos en investigación corresponden al software.



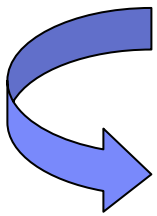
Software en el ocio: Efectos especiales

- **Poseidón:**

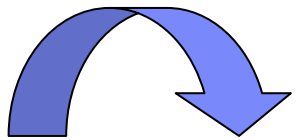
  El agua que aparece continuamente debe parecer real (pero no serlo, los actores deben servir para otras películas).



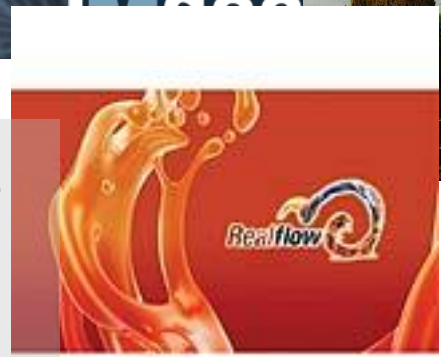
Esto es lo que se rueda, con un fondo en blanco



El programa simula el comportamiento del agua



Finalmente se le da el aspecto real



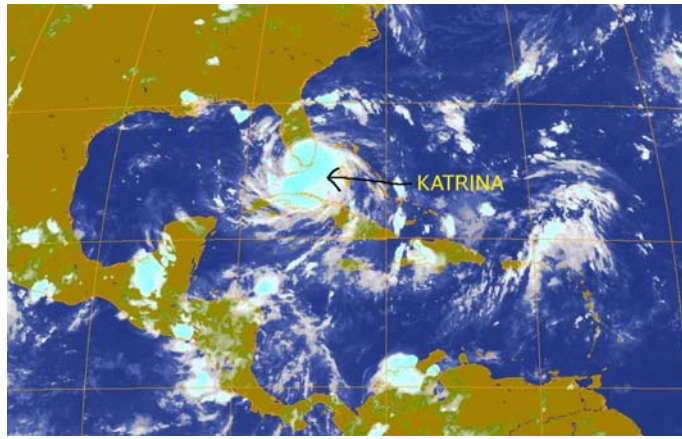
- Todos los efectos están creados por la empresa española Next Limit.
- Su producto simula todo tipos de fluidos.

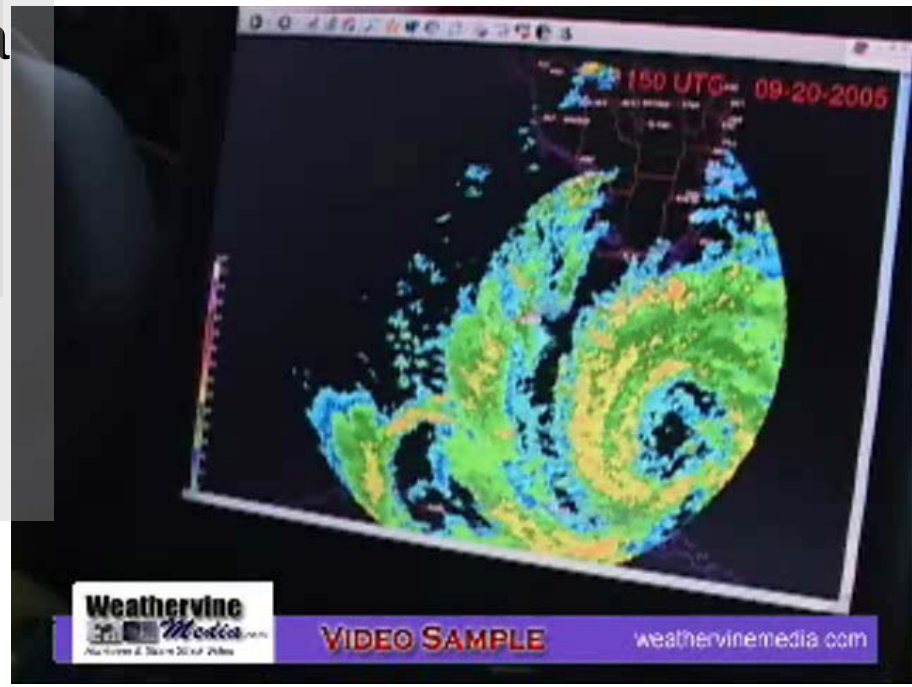
Software en el hogar

- Muchos de los aparatos que nos rodean en casa llevan software dentro: televisión, DVD, lavadoras, neveras.
- Dentro de muy poco las casas serán domóticas: el software controlará muchas de sus tareas.
- Se podrá acceder desde un computador o el móvil desde fuera de casas a funciones del hogar:
  - Ver las imágenes del videoportero ante una llamada.
  - Dar la orden de preparar la comida al salir del trabajo
  - Controlar la calefacción
  - Controlar las alarmas

Software y clima

- Entre los muchos modelos informáticos y simulaciones de los fenómenos físicos destaca el del clima.
  - Necesita comportamientos, por ejemplo el de un huracán o un tsunami, para prever su comportamiento
  - Además paralelización automática de modelos y optimización de recursos.





Software y otras ciencias

- Muchas otras ciencias basan sus modelos y avances en el software. Por ejemplo, necesitan modelos computaciones y simulaciones de estos modelos.

- Un ejemplo: Modelar el cerebro.
- Uno de los grandes retos de la humanidad: ¿lo completaremos alguna vez?
- Entre sus principales impulsores está Ramón y Cajal, premio Nóbel en 1906: Descubrió los mecanismos que gobiernan la forma y las conexiones de las neuronas.

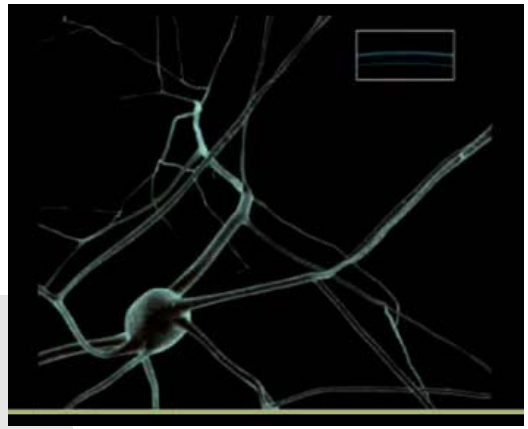
- Avances muy significativos:

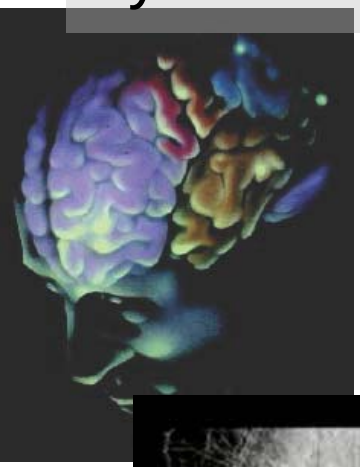
- Un pequeño ejemplo: un programa de computador que juegue al ajedrez como un campeón
- Conseguido en Mayo de 1970.

- Deep Blue ganó a Gary Kasparov: 3,5 a 2,5
- Deep Blue (el supercomputador RS/6000 SP) puede analizar 200.000.000 posiciones en un segundo, Kasparov 3.
- Pero Kasparov sabe mucho más de ajedrez, Deep Blue es principalmente fuerza bruta.

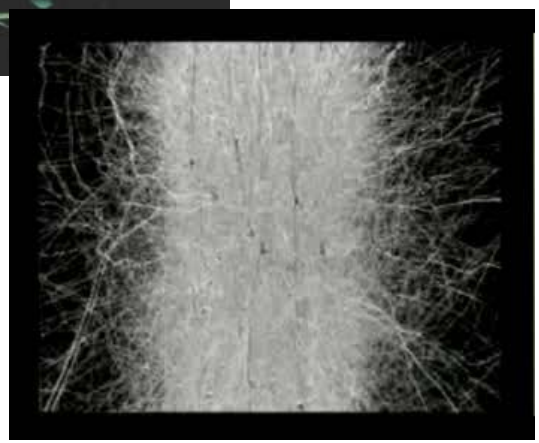
Software para modelar el cerebro

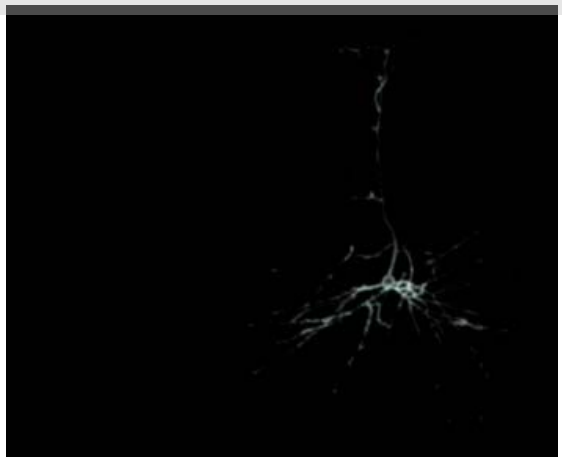
- Proyecto Blue Brain
- Desarrolla modelos de las neuronas y funciones cerebrales





- Simula su comportamiento.
- Comprender su funcionamiento
- Tratar enfermedades mentales





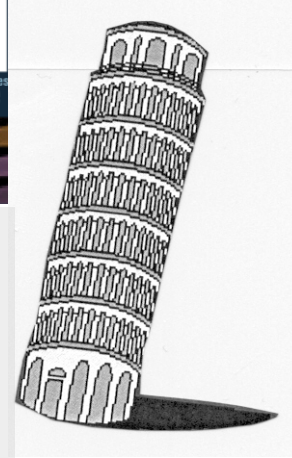
Software: motor económico y social

- El mercado del Software en el mundo ronda los 200 millardos €, un 1/3 en Europa creciendo 2% al año (en España crece un 8%)
- Los sistemas software están presentes en todo nuestro entorno y controlando aspectos importantes de nuestras vidas: sistemas bancarios, comunicaciones, transportes, medicina, etc. Todas las industrias apuestan por su conversión digital y el software es la clave.
- Es una de las principales fuentes del crecimiento económico: EEUU (el 30% de los últimos años, la industria del software da empleo a 3Mpersonas), un impacto similar en Finlandia, Israel e Irlanda.

Software: motor económico y social

2015	R&D expenses (Billion Euro)	Software R&D expenses (% of total R&D expenses)	Software R&D expenses (Billion Euro)	WW market size (Billion Euro)	Value added (Billion Euro)
Aerospace	51	45%	23	341	191
Automotive	129	35%	45	1,355	705
Consumer Electronics	21	60%	13	197	110
Medical Equipment	84	33%	28	471	280
Telecom Equipment	36	65%	23	257	144
Automation	3	15%	0.5	42	30
TOTAL	**€324 billion**		**€132 billion**	**€2,663 billion**	**€1,460 billion**

Problemas en el Software



Una gran parte del software producido sigue siendo de mala calidad.

- Parte de requisitos mal analizados: la validación es incorrecta.
- Costes de resolver fallos altísimos.
- Son problemas que conviven con el SW desde el principio, pero más importantes cuando nos vemos inmersos en el nuevo paradigma de los *Sistemas de Software Intensivo:*
  - Nuevas tecnologías emergentes basadas en sistemas empotrados
  - Parte de sistemas globales de cómputo (Internet, grids, computación orientada al servicio,...)

Fallos del Software

- **Ejemplos conocidos**
  - Lanzadera Ariane 501 (informe ESA)
  - Efecto 2000
  - Problema del Euro
  - Alarmas excesivas sobre electricidad en EEUU (First Energy infrastructure) con el resultado de 3 muertos
  - Accidente de un avión de los Marines americanos año 2000 (sin supervivientes)
  - Derribo de un avión amigo por un misil Patriot en Irak
  - Serios fallos en los sistemas de telecomunicaciones en Francia en 2004

Fallos del Software

- El problema de la fiabilidad como problema social: Artículo en *The Economist* (19 Junio 2003)

  "To achieve predictable quality in software-making, just as in carmaking; the more you automate the process, the more reliable it is".

  "The Panama incident causes some industry experts to consider the possibility that more stringent regulation of SW development is necessary"

  - Efectos de los métodos formales
    - Detección de errores en X-21 y TCP

El coste de los fallos

Un estudio del America's National Institute of Standards and Technology (NIST) 2002:

- Los fallos del SW son tan habituales que su coste es cercano a los 60 millardos de $ al año. Esto es un 0.6% del PIB.
- Un 80% de los costes del desarrollo de software se gastan en identificar y resolver fallos

NIST Study: Software Bugs Take Bite Out of Nation's Economy

El paradigma de los Sistemas SW Intensivos

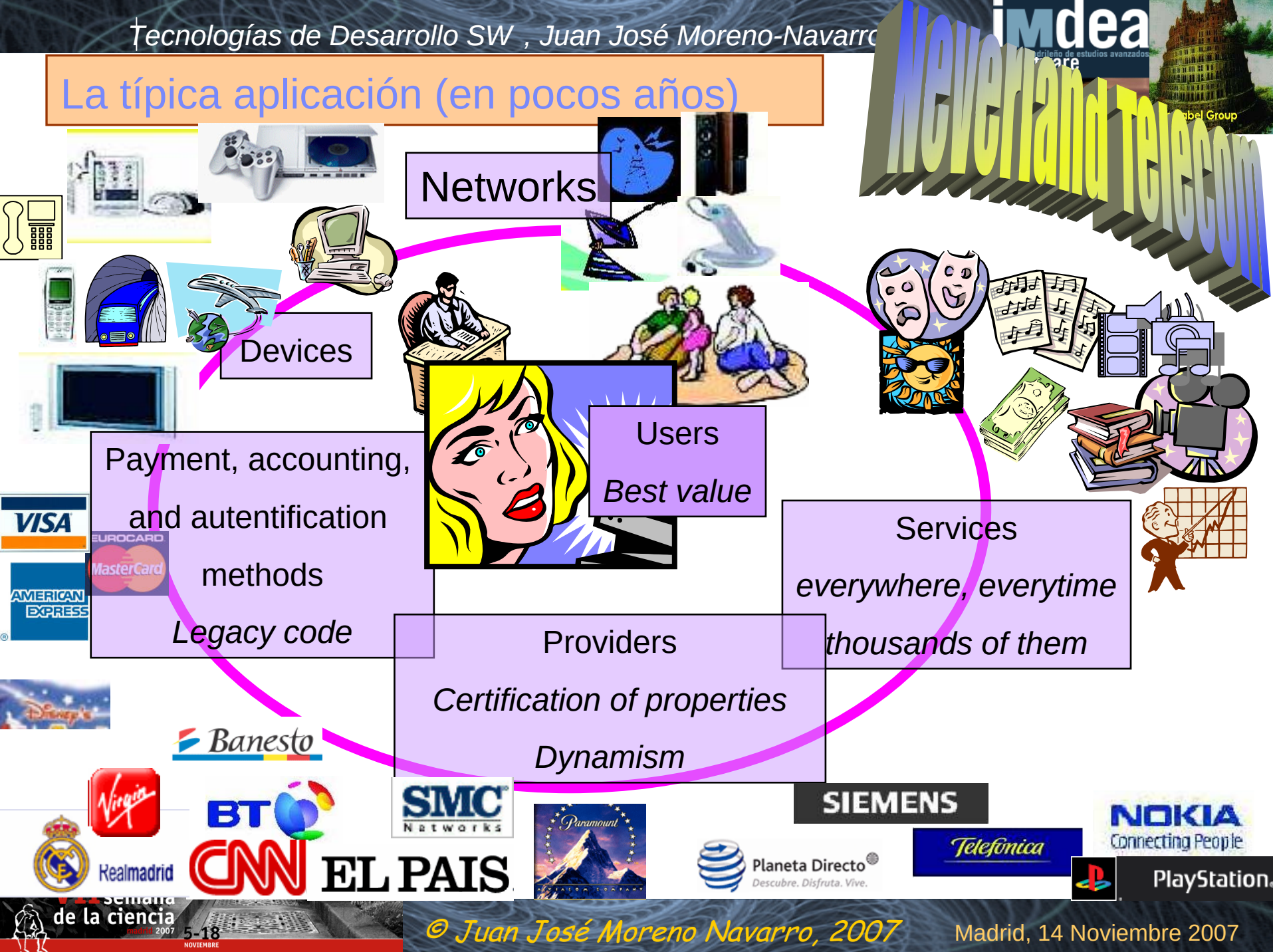
- Sistemas programables típicamente basados en sensores y varios controladores integrados.
- Distribuidos y descentralizados.
- Comportamiento adaptativo y anticipatorio, procesan conocimiento (y no solo datos) y cambian su estructura dinámicamente.
- Actuarán cómo computadores globales en entornos altamente dinámicos. Basados e integrados en el paradigma de software orientado a servicios.
- Servicios SW: entidades computacionales autónomas e independientes de la plataforma, que pueden describirse, publicarse, descubrirse y ensamblar automáticamente para desarrollar sistemas masivamente distribuidos, interoperables y con evolución controlada.

La validación del Software

La validación de los nuevos sistemas de software intensivo es más importante y difícil:

- Para los sistemas empotrados no podemos actualizar versiones o reparar errores en ejecución ya que el software se vende con bienes de larga duración (automóviles, aviones, satélites, teléfonos móviles, etc.)
- En el caso de una aplicación basada en servicios software, se ignora el código y el servicio solo puede usarse en operación (y probablemente pagando por ello).

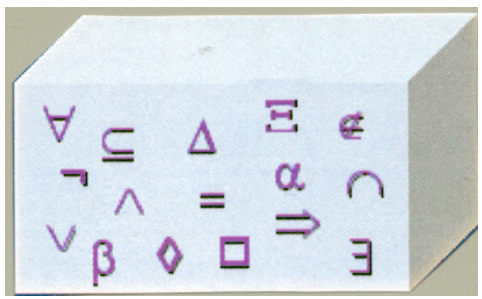
La típica aplicación (en pocos años)



El desarrollo de software en el futuro inmediato

- El software cada vez más complejo, difícil de desarrollar y validar/verificar “a mano”
- Requisitos no funcionales más importantes: Fiabilidad, seguridad, rendimiento, ...

¿Es posible la máquina para generar programas de alta calidad?

Requisitos →  → Programa correcto y de calidad

Desarrollo de software automatizado

¿Es posible la máquina para generar programas de alta calidad?

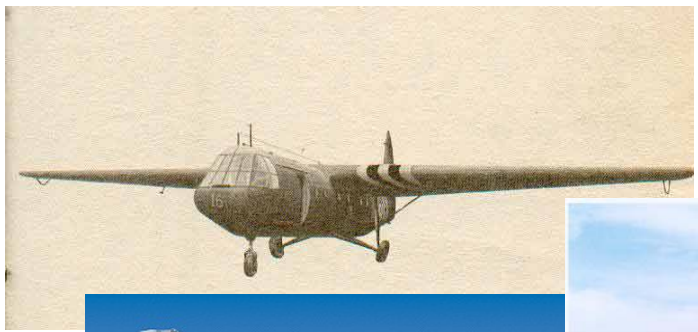
Nuestra respuesta es SÍ, a partir de tecnología rigurosa.

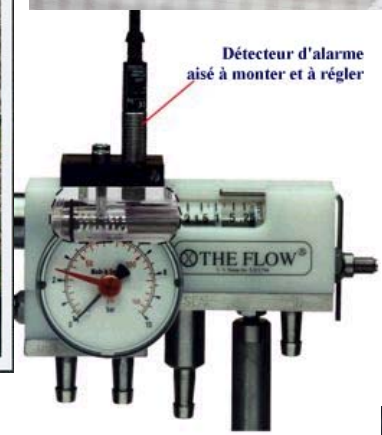
Esto engloba métodos, modelos, lenguajes y herramientas con una semántica rigurosa basados en el modo de razonar humano (*matemáticas /lógica)* :

- *Lenguajes de especificación, lenguajes de programación declarativos, métodos formales, verificación y validación formal, síntesis y transformación de programas, ...*

Ingeniería en el software

La ingeniería está llena de ejemplos donde se logra *fiabilidad*, a partir de una buena y rigurosa modelización de los problemas combinándolo con la experiencia de la producción.





Nuevos retos / nuevos modelos

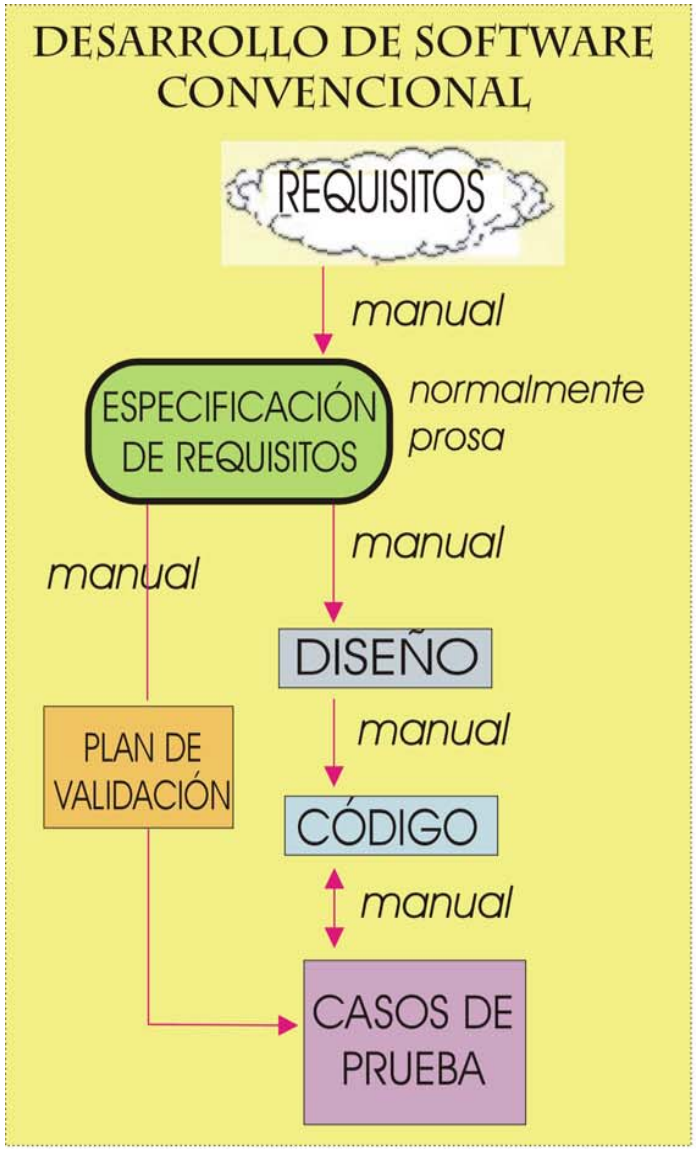
Parece imprescindible un cambio en los modelos

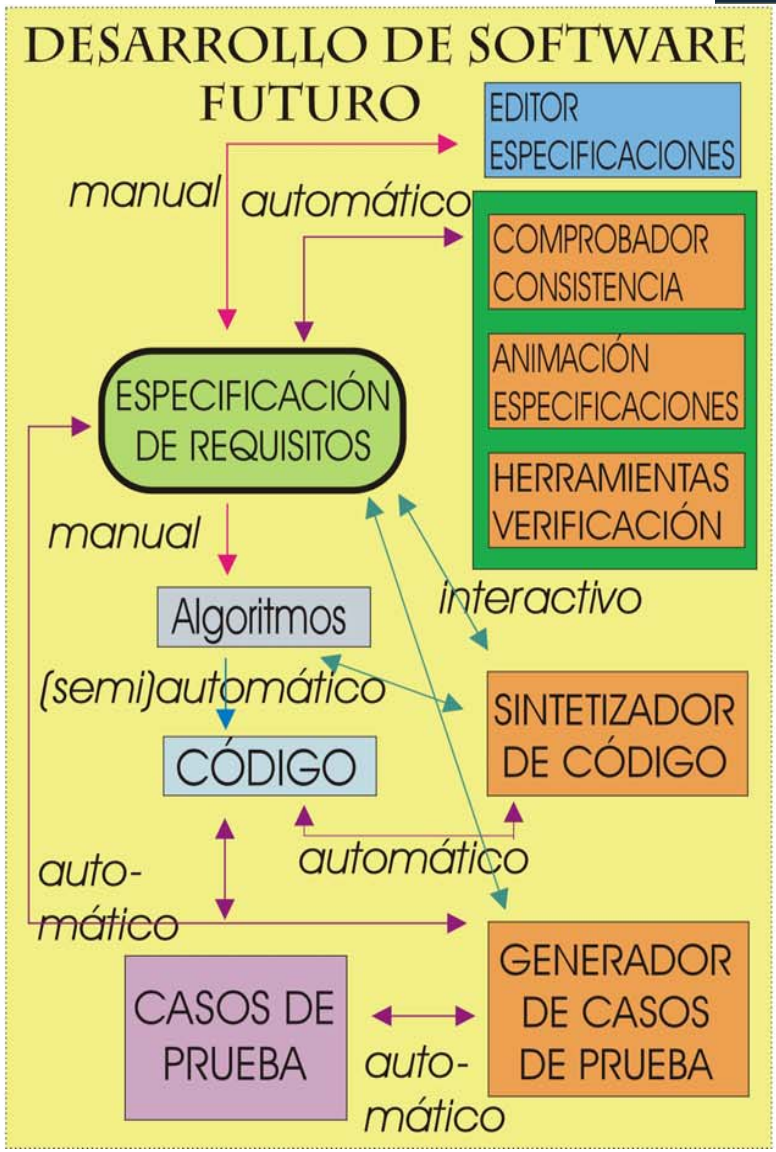
- *Uso únicamente de elementos semánticamente rigurosos en todas las etapas*
  - Análisis (Lenguajes de especificación)
  - Modelado (¿UML?)
  - Metodologías que aseguren corrección.
  - Herramientas que certifiquen propiedades.
  - Diseño (Lenguajes – formales - de descripción de arquitecturas, patrones de diseño [formalizados])
  - Implementación (Lenguajes de programación con semánticas claras: Declarativos, imperativos convenientemente documentados)

Nuevos retos / nuevos modelos

Principios básicos pero fundamentales

- Especificación rigurosa (incluso de los requisitos no funcionales)
- Procesos basados en métodos rigurosos (¿formales?)
- Herramientas que automaticen (parte de) el proceso
- Incrementar la reutilización / Interoperabilidad semántica / Documentación adecuada





Producción de una especificación de alta calidad del comportamiento del software

Generación de código eficiente y correcto

Atacando el problema de la calidad y fiabilidad

Dos maneras de abordar el problema

1. Proporcionar herramientas para comprobar propiedades del código existente (model checking, interpretación abstracta, herramientas de validación [generadores de pruebas, ...], herramientas de verificación [demostradores de teoremas], ...).
2. *Corrección por construcción*: Proporcionar una metodología que genere código correcto desde las primeras etapas del desarrollo.

Nuevos métodos de desarrollar SW

- Cada vez hay más y más proyectos desarrollados con tecnología rigurosa.
- Cada vez hay más herramientas permitiendo su aplicación.
- Los métodos más formales se ocultan al usuario dentro de las herramientas.
- ¿Se imagina uno una ingeniería sin formalismos?

Nuevos métodos de desarrollar SW

Uno de los mitos de estos métodos: *Son más costosos*

- Reutilización: razones económicas, de fiabilidad y calidad. Potencialmente universal si los componentes están bien Esperanzas especificados y certificados y se proveen herramientas que faciliten la búsqueda (semántica y automática) de servicios y componentes.

- El coste de la certificación (por ejemplo, con técnicas formales) de un servicio software que va a utilizarse miles de veces puede ser insignificante cuando se factura ... y altamente rentable.

Nuevos métodos de desarrollar SW

Algunos ejemplos:

- Model checking en los drivers de windows en Microsoft (proyecto slam).
- Atelier B
- Trusted Logic: Certificación de protocolos criptográficos.
- Spark: Herramientas de validación para Ada. Aplicaciones en el campo de la seguridad y certificación
- Precise UML/OCL: Herramientas de comprobación, Thales.

Un ejemplo relevante

SW para el sistema de control primario de vuelo del A340/A380

- Programa en C, generado manualmente a partir de una especificación de alto nivel.
- A340: 75.000 líneas, 10.000 variables globales
- A380: × 3

- Validación con técnicas y SW convencional: 3.5 días, detecto errores pero 4.200 falsas alarmas.
- Con interpretación abstracta: 40 minutos/7 horas, detectó todos los errores anteriores, 0 falsas alarmas
- *Patrick Cousot, ENS París, Sistema Astree*

Otro ejemplo relevante

Línea 14 del metro de París

- Completamente automatizada (tráfico y trenes)
- Completamente especificada en B
  - 100.000 líneas de especificación incluyendo todos los refinamientos
  - 87.000 líneas de código Ada (semiautomático)
  - 27.000 pruebas (semiautomático)

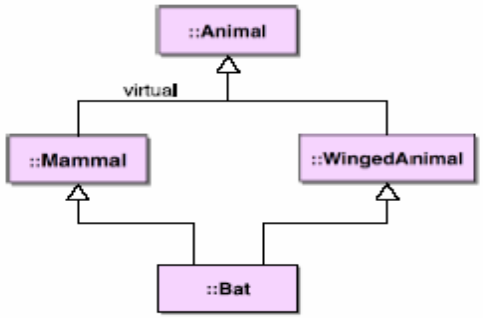
Ningún error al validarlo con técnicas convencionales

¡Todavía en uso la versión 1.0 desde Octubre 1998!

Y otro más: GCCC

Knowledge Base About a Program

A set of classes violating rule HICPP 3.3.15:

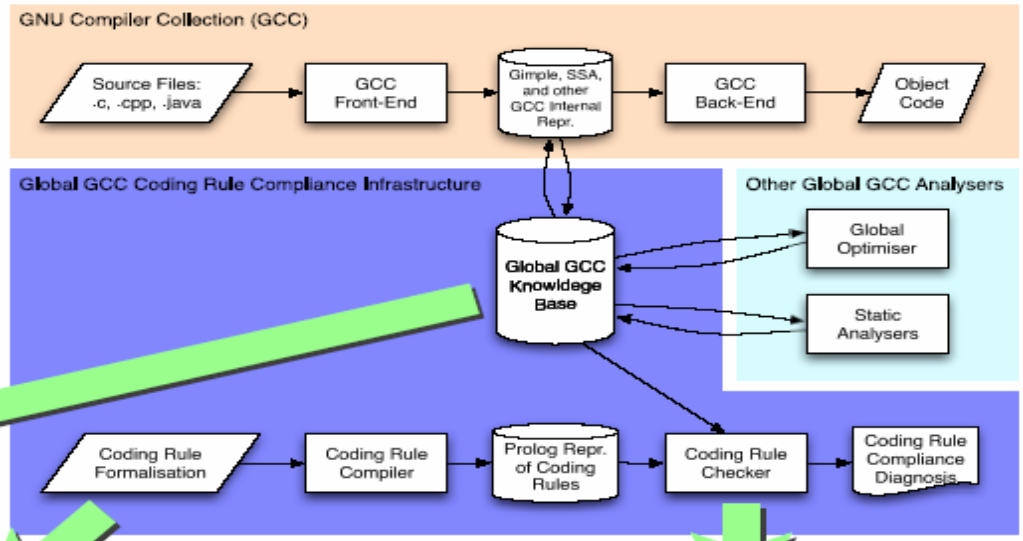


Program properties gathered during compilation for the above example and relevant to rule HICPP 3.3.15:

```
class('::Animal').
class('::Mammal').
class('::WingedAnimal').
class('::Bat').
direct_base_of('::Animal','::Mammal').
direct_base_of('::Animal','::WingedAnimal').
direct_base_of('::Mammal','::Bat').
direct_base_of('::WingedAnimal','::Bat').
virtual_base_of('::Animal','::Mammal').
```

The Global GCC Project

- ITEA funded (2006-08) consortium of industrial / research partners
- Goal: improving static analysis capabilities of the GNU Compiler Collection (GCC)
- **Global GCC knowledge base:** share information among different GGCC analysers



Rule Formalisation

Based on **first-order logic** and written in a domain-specific language which is translated into Prolog and that:

- Formalises standard coding rule sets in a declarative style
- Makes it easier for the final user to define additional coding rules
- Provides a collection of predefined predicates about program facts (such as `class/1`, `base_of/2`, or `in_call_graph_of/2`)
- Quantification over certain domains
- Constructive negation

Rule Checking

Rule HICPP 3.3.15 translated into Prolog

```
violate_hicpp_3_3_15(A, B, C, D) :-
      class(A), class(B), class(C), class(D), B \= C,
      direct_base_of(A, B), direct_base_of(A, C),
      base_of(B, D), base_of(C, D),
      \+ virtual_base_of(A, C).
```

We do not code the rule itself, but **its negation**. Any program that satisfies the negated rule thus violates the coding rule.

Predicates coding rule violations are queried against facts describing a program. A successful resolution flags a rule violation, providing a witness.

GCCC

PROJECT / RULE \ KLOC	Bacula 20	CLAM 46	Firebird 439	IT++ 39	OGRE 153	Orca 89	Qt 595
3.3.1	0 (0)	1 (0)	16 (0)	0 (0)	0 (0)	1 (0)	9 (0)
3.3.2	3 (0.03)	15 (0.64)	58 (1.07)	6 (0.04)	10 (0.56)	12 (0.17)	76 (11.11)
3.3.15	0 (0)	0 (0.16)	0 (0.2)	0 (0)	0/1 (0.09)	0 (0.12)	4/5 (1.16)

[no. of true violations /] no. of violations found (running time

PROJECT	DESCRIPTION
Bacula	A network based backup program.
CLAM	C++ Library for Audio and Music.
Firebird	Relational database with concurrency and stored procedures.
IT++	Library for signal and speech processing.
OGRE	Object-Oriented 3D Graphics Rendering Engine, scene-oriented.
Orca	Framework for developing component-based robotic systems.
Qt	Application development framework and GUI widgets library.

IMDEA Software

- Un instituto de la Comunidad de Madrid centrado en las *Tecnologías de Desarrollo de Software*.
- Integrado con las Universidades: UPM, URJC, UCM, CSIC
- Clara vocación de excelencia científica

IMDEA Software

- Centro de referencia en Europa
- El primer centro de investigación en Software en España (y uno de los 3 en Informática)
- 100 investigadores de prestigio en 5 años
- Conexión con las empresas: Telefónica I+D, Atos Origin, BBVA
- Sede provisional: Facultad de Informática
- Sede definitiva: Nuevo edificio en el Parque Tecnológico de Montegancedo

IMDEA Software

- Nuevos investigadores en los próximos meses.
- Idioma de trabajo: Inglés
- Convocatoria de investigadores y de contratos doctorales en curso: www.imdea.org/software

NESSI & INES

- Plataformas Tecnológica sobre Software y Servicios (Europa y España)
- Red de cooperación entre agentes

- UPM/IMDEA miembro activo de ambas

Making predictions is difficult, specially for the future

Niels Bohr (Premio Nóbel de Física, 1922)

Si Dios hubiera querido que el hombre volara le hubiera dado alas y otra cosa es una blasfemia. Volar está reservado a los pájaros y a los ángeles

Obispo Milton Wright
Padre de los hnos. Wright

La mejor manera de predecir el futuro es inventarlo

Alan Kay / Expediente X

Gracias por su atención