

Supongo que como yo, algunas veces piensas como sería el mundo si las máquinas pensaran. De existir, yo me imagino **la máquina pensante** como la unión perfecta entre las capacidades *“pensantes”*
de los humanos y las capacidades computacionales de los ordenadores.

El tema da para mucho, pero comentemos algo que debería inquietarte como programador: **¿p odrán las máquinas sustituir a los programadores?**
.

Ésto nos lleva indefectiblemente a la siguiente, y mucho más pragmática (y útil), pregunta: **¿co mo será el programador del futuro?**
.

“La máquina pensante” o “El hombre máquina”

El futuro, y más siendo lejano, es incierto. Más importante que si acertamos o no (que para entonces, nos dará lo mismo) es conocer mejor el mundo en que vivimos hoy (informático en este caso).

Hasta donde yo se (que al respecto no es mucho) nadie tiene ni la más remota idea de qué es la inteligencia. Se cree, se dice, se comenta, … bla, bla, bla, … ni idea. Las definiciones que se dan comúnmente no definen la inteligencia, sino que la describen, pero dejémoslo aquí, que nos vamos del tema.

Las únicas dos vías que yo conozco para reproducir artificialmente la inteligencia son: por un lado, simular el comportamiento fisiológico de nuestro cerebro (donde **se cree** que reside/proviene), y por otro lado, reproducir sintéticamente un comportamiento inteligente (predecir el tiempo que hará mañana **parece** de inteligentes, pero un ordenador únicamente realiza billones de operaciones de la forma más tonta posible).

Del primer método, nadie sabe qué resultará: si una especie de **savant** sociópata, un autómata como cualquier otro, un gasto inútil, etc… Es más, algunos expertos indican que será al revés, que los humanos obtendremos las capacidades de los ordenadores mediante implantes, evolucionando hacia **"El hombre máquina"**. Si es lo primero, parece que tardará bastante y si es lo segundo ¡nos ponemos el implante!.

Del segundo método, **sí hay que preocuparse**. Debemos reconocer que muchas de las tareas que tradicionalmente se han considerado *"de inteligentes"*, las realizan cada vez mejor y en mayor cuantía los ordenadores (eg. recomendar productos según los gustos personales, clasificar las fotos en las que aparece Pepe, conducir un coche, etc…). Pero, ¿que hay de la inteligencia que poseemos los programadores?, ¿hasta que punto esta inteligencia *"sintética"* nos relevará a corto plazo?, ¿o acaso ya está ocurriendo?.

El método sintético y sus limitaciones

El método sintético para lograr inteligencia artificial ha sido denostado por algunos que creían (eg. *"2001: Una odisea del espacio"* en 1968, *"Juegos de Guerra"* en 1983 y tantas otras) que en pocos años los ordenadores nos mirarían a los ojos. Pero se ha demostrado que el método funciona y que nos aporta inteligencia extra a los programadores menos inteligentes.

Desde el punto de vista del programador, inteligencia artificial es *“cualquier habilidad (del ordenador, claro) que permita hacer, a un*
programador menos preparado
, lo que sin esa habilidad requiere un
programador más preparado
“
.

No hace demasiados años (eg. **OpenGL** apareció en 1992, pero yo aún recuerdo trabajar con un tortuoso manual de las Glide para 3dfx que nació en 1994), quien quería hacer algún programa con gráficos debía tener cristalinos conocimientos de la plataforma hardware, de ensamblador, de las características de las diferentes tarjetas gráficas, de los algoritmos de trazado, soltura con la geometría, etc… ¿cuantos de los presentes han tenido que asignar un valor al registro **DS** para poder cambiar de página al dibujar en resoluciones de 640×320 en una CGA, EGA o VGA? (sí, ya se que tú sí, pero he preguntado cuantos).

Hoy no, hoy los ordenadores realizan de forma inteligente (fijaros si será de forma inteligente que muchos programadores no sabríamos hacerlo nosotros mismos):

- Optimizaciones eficientes a nivel de código máquina.
- Optimización eficiente de bases de datos en base a estadísticas de su uso (tanto de las consultas como de los índices y objetos para optimizarlas).
- Recomendaciones y correcciones constantes en la escritura de código (el de Visual Studio siempre ha sido muy bueno, pero el de IntelliJ para Scala está francamente bien).

- **Todo** el aparato hardware se ha abstraído exitosamente, cualquier programador hoy en día (claro, los que no habéis programado *“en el pasado”* lo veis natural) puede imprimir, escanear, grabar audio, vídeo, etc… independientemente del hardware y **trivialmente**
(a alguno os sonará eso de, en los juegos, configurar la tarjeta gráfica [CGA, EGA, VGA] y la de sonido [Sound Blaster for ever]). Me acuerdo de las primeras impresoras que me compré ¡de 9 agujas! que en el manual venía ¡el repertorio de códigos para programarla!.

- Recolección de basura. Sí, ya da igual si eres un chapucero que no sabes cuando liberar la memoria ¡el ordenador detecta cuando ya no se usa y lo hace por tí!.
- Matemática binaria (eg. convertir *“x / 2”* en un *“x >> 1”*), desenrollado automático de bucles, paralización automática (eg.
“Tareas.AsParallel().Algo(…)”

), ya da igual si el programador
no sabe hacerlo
, la máquina lo hace por él.

- La capacidad de poder re-usar el código de otros (utilizar la inteligencia de otros), permite que lo que sólo unos pocos pueden hacer (eg. resolver SAT con coste *"O(2^0.41n)"*, reconocimiento de voz, tomografía computarizada, etc…) esté al alcance de cualquiera, independientemente de sus conocimientos.

- etc…

Repito, que no te confunda el hecho de que algunas cosas no te parezcan inteligentes. Si el ordenador, es capaz de hacer cosas (y todas las anteriores lo son) que un programador **no puede hacer**
porque no tiene conocimientos o inteligencia suficiente, es que el ordenador es inteligente (eg. un ordenador
no es
más inteligente que un programador porque el primero sume dos números en 10^{-9}
segundos y el segundo necesite papel y lápiz, lo es porque el ordenador es capaz de deducir que una instrucción no hace falta y puede ser optimizada ¡cuando el programador cree que sí hace falta!).

Y sí, los avances en teoría de lenguajes, computación y complejidad computacional permiten intuir un futuro (aún algo lejano, afortunadamente) en el que la máquina podrá **analizar, comprobar y reducir un teorema**
(ver
ATP
en referencias) lo que implica, que
dará igual que un programador escriba un código pésimo
, bastará con que sepa indicar al ordenador sus intenciones.

La cuestión, es que aunque el programador sea un torpe y escriba un código horrorosamente ineficiente ¡el ordenador generará la solución con coste constante! (la más eficiente). **¿Para qué quiero pagar a un programador inteligente si con uno que**
"se apañe"
obtendré el mismo resultado?
.

La brecha

Sí, todo lo comentado anteriormente ya está aquí (bueno, lo del **ATP** no, afortunadamente). Cada vez más gente no especializada puede realizar programas que hasta hace poco, sólo estaba al alcance del programador experimentado (Internet está hasta arriba de ellos).

¡Pero si eso es bueno, la "Universalización de la programación"!, ¿no?.

Depende para quien, para el programador al que le gusta optimizar su código en ensamblador no es nada bueno, porque no podrá mejorar a un ordenador que será capaz de obtener, en mili-segundos, **la mejor** solución (mala suerte **Michael Abrash**).

Como resultado, y si nos fijamos en situaciones similares, se abrirá una brecha entre la élite de la programación (que investigarán) y el resto de los mortales a los que cada vez más costará poseer una cualidad diferenciadora (hablando de programación). Si la brecha será muy profunda (todo el mundo juega al fútbol, pero sólo unos pocos son los elegidos para la gloria) o no tan profunda (los médicos, jueces y notarios lo tienen bien montado) sólo el tiempo lo dirá.

¿Entonces?, ¿cuelgo el teclado?

"La brecha" actualmente es perceptible pero no muy pronunciada, tenemos margen para enarbolar nuestras aptitudes diferenciadoras. No obstante, las aptitudes diferenciadoras tradicionales de un programador se transformarán (ya lo están haciendo) en otras aptitudes diferentes y a esas es a las que hay que estar atento. Un ejemplo claro es que antes un programador era valorado por la cantidad (y calidad) del código que escribía (sí, ya, COCOMO, no hablo de eso). Sin embargo ahora, dada la ingente cantidad de código **que ya hay escrito**, un programador que sepa buscar y combinar (estilo *"mashup"*) el código de otros, tendrá más éxito hoy en día ¡y siendo peor programador que aquellos de los que usa el código!.

Hay otras muchas aptitudes que nos mantendrán *en la cresta de la ola* (como ser buenos analistas/arquitectos). Pero de la misma forma que la moda de optimizar en ensamblador ha pasado (no, no, ya no es lo de antes, ¡antes era vicio!), la moda de escribir buen código pasará (ya hay mucho mal código por ahí que triunfa por motivos ajenos a la programación [de Windows siempre lo han dicho…]) y eso es malo para los que nos gusta programar.

La singularidad tecnológica

Llevado al extremo, hay quien piensa que llegará el momento en que la inteligencia artificial (en cualquiera de sus formas) evolucionará tan deprisa que nos dejará rápidamente atrás. A ese momento lo llaman la **Singularidad tecnológica** (ver referencias).

Hasta que llegue ese momento, yo seguiré soñando con el algoritmo de Dios.

Más información | Singularidad tecnológica , Demostración automática de teoremas (ATP), Algoritmo de Dios
Más información | Michael Abrash especialista en optimizar código, Mashu

Fuente: http://www.genbetadev.com/formacion/el-programador-del-futuro