Sobre la crueldad de verdaderamente enseñar ciencias de la computación

Edsger W. Dijkstra.

La segunda parte de esta charla denota algunas de las consecuencias científicas y educacionales provenientes de la idea de que las computadoras representan una novedad radical. Para darle contenidos claros a esta asunción, tenemos que ser mucho más precisos acerca de lo que queremos decir en este contexto con el uso del adjetivo "radical". Lo haremos en la primera parte de esta charla, en la cual vamos a proveer evidencia que respalde nuestra suposición.

La manera usual en la cual hoy planificamos para el mañana es en el vocabulario de ayer. Lo hacemos porque tratamos de avanzar con los conceptos que nos son familiares, los cuales han adquirido un significado en nuestra experiencia pasada. Por supuesto, las palabras y los conceptos no encajan precisamente porque nuestro futuro difiere de nuestro pasado, pero las estiramos un poco. Los lingüistas están bastante familiarizados con el fenómeno en el cual los significados de las palabras evolucionan a través del tiempo, pero también saben que este es un proceso lento y gradual.

Es el método más común cuando se trata de lidiar con la novedad: utilizando metáforas y analogías tratamos de vincular lo nuevo con lo viejo, lo novedoso con lo familiar. Bajo un cambio suficientemente lento y gradual, esto funciona razonablemente bien; en el caso de una discontinuidad aguda, sin embargo, el método colapsa: aunque podemos glorificarlo con el nombre "sentido común", nuestra experiencia pasada ya no es más relevante, las analogías se tornan muy superficiales, y las metáforas se hacen engañosas en vez de reveladoras. Esta es la situación que caracteriza a la novedad "radical".

Lidiar con una novedad radical requiere un método ortogonal. Uno debe considerar su propio pasado, las experiencias recogidas, y los hábitos formados en él como un desafortunado accidente de la historia, y debe acercarse a la novedad radical con la mente en blanco, rechazando conscientemente el intento de vincularla con lo que ya es familiar, debido a que lo familiar es desesperanzadamente inadecuado. Uno debe, con una especie de personalidad dividida, tomar una novedad radical como algo desasociado por propio derecho. Comprender una novedad radical implica crear y aprender un lenguaje extraño, el cual no puede ser traducido a nuestra lengua materna. (Cualquiera que haya aprendido mecánica cuántica sabe a lo que me refiero.) No hace falta decirlo, ajustarse a las novedades radicales no es una actividad muy popular, ya que requiere mucho trabajo. Por la misma razón, las novedad radicales no son por sí mismas bienvenidas.

A esta altura, bien se pueden preguntar por qué he prestado tanta atención y he gastado tanta elocuencia en una noción tan simple y obvia como una novedad radical. Mi razón es muy simple: las novedades radicales son tan perturbantes que tienden a ser suprimidas o ignoradas, al punto que la mera posibilidad de su existencia es generalmente negada antes que admitida.

Voy a ser breve en cuanto a la evidencia histórica. Carl Friedrich Gauss, el Príncipe de los Matemáticos pero algo cobarde, sin duda estaba al tanto del destino de Galileo –y probablemente podría haber predicho las acusaciones a Einstein– cuando decidió suprimir su descubrimiento de la geometría no Euclidiana, dejando que Bolyai y Lobatchewsky recibieran las críticas. Es probablemente más revelador ir un poco más atrás, a la Edad Media. Una de sus características era que "razonar mediante analogías" era descontrolado; otra característica era el total estancamiento intelectual, y ahora vemos porque ambas

características van juntas. Una razón para mencionar esto es resaltar que, desarrollando un oído entrenado para las analogías no garantizadas, uno puede detectar una gran cantidad de pensamiento medieval hoy en día.

La otra cosa que no puedo resaltar lo suficiente es que la fracción de la población para la cuál el cambio gradual parece ser cualquier cosa menos el único paradigma de la historia es muy grande, probablemente mucho más grande de lo que esperarían. Ciertamente cuando comencé a observarlo, su número resultó ser mucho mayor de lo que esperaba.

Por ejemplo, la gran mayoría de la comunidad matemática nunca ha confrontado la suposición tácita de que hacer matemáticas va a continuar siendo básicamente el mismo tipo de actividad mental que siempre ha sido: los nuevos temas vendrán, florecerán, e irán como lo han hecho en el pasado, pero siendo lo que es el cerebro humano, nuestras formas de enseñar, aprender, y el entendimiento las matemáticas, la resolución de problemas y el descubrimiento matemático van a continuar siendo básicamente lo mismo. Herbert Robbins expone claramente por qué él descarta un salto cuántico en la habilidad matemática:

> "Nadie va a correr 100 metros en cinco segundos, sin importar cuánto se invierta en entrenamiento y máquinas. Lo mismo puede decirse acerca del uso del cerebro. La mente humana no es diferente ahora de lo que era hace cinco mil años. Y cuando se trata de matemáticas, debe darse cuenta de que se trata de la mente humana a un extremo límite de su capacidad".

Mi comentario en el margen fue "¡entonces reduzca el uso del cerebro y calcule!". Usando la propia analogía de Robbins, uno puede resaltar que, para ir rápido desde A hasta B, podrían existir ahora alternativas a la de correr que son órdenes de magnitud más efectivas. Robbins rechaza de llano honrar cualquier alternativa al valioso uso del cerebro llamado "hacer matemáticas", exorcizando así el peligro de la novedad radical mediante el simple método de ajustar sus definiciones a sus necesidades: simplemente por definición, las matemáticas continuarán siendo lo que solían ser. Demasiado para los matemáticos.

Déjenme darles un ejemplo más de la desconfianza generalizada sobre la existencia de las novedades radicales y, por consiguiente, de la necesidad de aprender cómo lidiar con ellas. Es el accionar educacional que prevalece, para el cuál el cambio, casi imperceptible, parece ser el paradigma exclusivo. ¡Cuántos textos educacionales no son recomendados porque apelan a la intuición del estudiante! Constantemente tratan de presentar todo aquello que podría ser una emocionante novedad como algo tan familiar como sea posible. Conscientemente tratan de vincular el material nuevo con lo que se supone es el mundo familiar del estudiante. Ya empieza con la enseñanza de la aritmética. En vez de enseñar 2 + 3 = 5, el horrendo operador aritmético "más" es cuidadosamente disfrazado llamándolo "y", y a los pequeños niños se les dan muchos ejemplos familiares primero, con objetos claramente visibles como manzanas y peras, que lo son, en contraste al uso de objetos numerables como porcentajes y electrones, que no lo son. La misma tonta tradición es reflejada a nivel universitario en diferentes cursos introductorios de cálculo para los futuros físicos, arquitectos, economistas, cada uno adornado con ejemplos de sus respectivos campos. El dogma educacional parece ser que todo está bien siempre y cuando el estudiante no se dé cuenta de que está aprendiendo algo verdaderamente nuevo; generalmente, el presentimiento del estudiante es de hecho correcto. Considero como un serio obstáculo la falencia de una práctica educativa en preparar a la próxima generación para el fenómeno de novedades radicales. [Cuando el Rey Fernando visitó la conservadora universidad de Cervera, el Rector orgullosamente aseguró al monarca con las palabras: "Lejos esté de nosotros, Señor, la peligrosa novedad de pensar". Los problemas de España en el siglo que siguió justifican mi caracterización del problema como "serio"]. Demasiado para la adopción por parte de la

educación del paradigma de cambio gradual.

El concepto de novedades radicales es de importancia contemporánea ya que, mientras estamos mal preparados para lidiar con ellas, la ciencia y la tecnología no se han mostrado expertas en influirlas sobre nosotros. Ejemplos científicos antiguos son la teoría de la relatividad y la mecánica cuántica; ejemplos tecnológicos modernos son la bomba atómica y la píldora. Durante décadas, los primeros dos ejemplos dieron lugar a un torrente de corrientes religiosas, científicas o de otra manera cuasi-científicas. Día a día podemos observar el profundo error de enfoque con el cuál los últimos dos ejemplos han sido abordados, ya sea por nuestros políticos y líderes religiosos o por el público en general. Demasiado para el daño hecho a nuestra paz mental por las novedades radicales.

Traje esto a colación debido a mi convencimiento de que las computadoras automáticas representan una novedad radical y de que sólo identificándolas como tal podemos identificar todo lo irrelevante, los conceptos errados y la mitología que las rodea. Una inspección más a fondo revelará que esto es todavía peor, a saber, que las computadoras automáticas engloban no sólo una novedad radical sino dos de ellas.

La primera novedad radical es una consecuencia directa del poder bruto de las computadoras actuales. Todos sabemos como lidiar con algo tan grande y complejo; divide y vencerás, por ejemplo vemos el todo como una composición de partes y tratamos con las partes por separado. Y si una parte es muy grande, repetimos el procedimiento. La ciudad está compuesto por barrios, que están a su vez estructurados por calles, que contienen edificios, que están hechos de paredes y pisos, que están construidas de ladrillos, etc. eventualmente llegando a las partículas elementales. Y tenemos a todos nuestros especialistas sobre el tema, desde el ingeniero civil, pasando por el arquitecto hasta el físico de estado sólido y consiguientes. Ya que, en cierto sentido, el todo es "mas grande" que sus partes, la profundidad de una descomposición jerárquica es algún tipo de logaritmo del cociente entre los "tamaños" del todo y las partes más pequeñas. Desde un bit a cien mega bytes, desde un microsegundo a media hora de cómputos nos confronta con un cociente completamente abrumador de 10^9! El programador está en la posición inigualada en la cual la suya es la única disciplina y profesión donde un cociente tan gigante, lo cual completamente sobrepasa nuestra imaginación, debe ser consolidado por una sola tecnología. Debe poder pensar en términos de jerarquías conceptuales que son mucho más profundas que todas aquellas que debió enfrentar una sola mente alguna vez. Comparado con ese número de niveles semánticos, la teoría matemática promedio es casi plana. Evocando la necesidad de profundas jerarquías conceptuales, la computadora automática nos confronta con un radical desafío intelectual que no tiene precedente histórico.

Nuevamente, debo enfatizar esta novedad radical ya que el verdadero creyente en el cambio gradual y las mejoras incrementales no puede verla. Para él, una computadora automática es algo como una familiar caja registradora, sólo que algo más grande, rápida y más flexible. Pero la analogía es ridículamente superficial: es órdenes de magnitud peor que comparar, como un medio de transporte, el avión supersónico con un bebé que gatea, ya que el cociente de velocidad es sólo de mil.

La segunda novedad radical es que la computadora automática es nuestro primer dispositivo digital de gran escala. Tuvimos un par de notables componentes discretos: Acabo de mencionar la caja registradora y podemos agregar la máquina de escribir con sus teclas individuales: con un sólo golpe podemos escribir puedes escribir una Q o una W pero, aunque las teclas están una al lado de la otra, no una mezcla de las dos. Pero tales mecanismos son la excepción, y la amplia mayoría de nuestros mecanismos son vistos como dispositivos analógicos cuyo comportamiento sobre un amplio rango es una función continua de todos los parámetros involucrados: si presionamos la punta del lápiz un poco más fuerte, obtenemos una

línea levemente más gruesa, si el violinista ubica su dedo levemente fuera de su posición correcta, reproduce una nota levemente desafinada. A esto debería agregar que, al punto que nos vemos como mecanismos, nos vemos primordialmente como dispositivos analógicos: si nos esforzamos un poco más esperamos rendir un poco más. A menudo el comportamiento no es solamente una función continua sino también monótona: para ver si un martillo es adecuado sobre un cierto rango de clavos, lo probamos con el más pequeño y el más grande de los clavos del rango, y si el resultado de ambos experimentos es positivo, estamos perfectamente predispuestos a creer que el martillo será apropiado para todos los clavos intermedios.

Es posible, inclusive tentador, ver a un programa como un mecanismo abstracto, como alguna clase de dispositivo. Pero hacerlo, sin embargo, es altamente peligroso: la analogía es muy superficial debido a que un programa es, como mecanismo, totalmente diferente de todos los familiares dispositivos analógicos con los cuáles crecimos. Como toda la información digitalizada, tiene la inevitable e incómoda propiedad de que la menor de las posibles perturbaciones –por ejemplo cambios a un sólo bit– puede tener las más drásticas consecuencias. [Por completitud agrego que la situación no cambia en su esencia por la introducción de la redundancia o la corrección de errores]. En el mundo discreto de la computación, no hay métrica significativa en la cual "pequeños" cambios y "pequeños" efectos vayan de la mano, y nunca los habrá.

Esta segunda novedad radical comparte el destino usual a todas las novedades radicales: es negada, porque su verdad sería demasiado incómoda. No tengo idea lo que esta negación y descreencia específica le cuesta a los Estados Unidos, pero un millón de dólares al día parece una modesta estimación.

Habiendo descripto –en los términos más amplios posibles, lo admito– la naturaleza de las novedades computacionales, debo ahora proveer la evidencia de que tales novedades son, de hecho, radicales. Lo haré explicando una serie de fenómenos que de otra manera serían extraños por la frustrante –pero, como ahora sabemos, condenada– ocultación o negación de su aterradora extrañeza.

Cierta cantidad de estos fenómenos han sido agrupados bajo el nombre de "Ingeniería de Software". Así como la economía es conocida como "La Ciencia Miserable", la ingeniería de software debería ser conocida como "La Disciplina Condenada", condenada porque ni siquiera puede acercarse a su meta, dado que la misma es en sí misma contradictoria. La ingeniería de software, por supuesto, se presenta a sí misma como otra causa valiosa, pero es un colirio: si lee cuidadosamente su literatura y analiza lo que realmente hacen quienes se avocan a ella, descubrirá que la ingeniería de software ha adoptado como su estatuto "Cómo programar si usted no puede".

La popularidad de su nombre es suficiente para hacerla sospechosa. En lo que denominamos "sociedades primitivas", la superstición de que conocer el verdadero nombre de alguien otorga un poder mágico sobre él no es inusual. Difícilmente somos menos primitivos: ¿por qué persistimos en contestar el teléfono con el poco útil "hola" en vez de nuestro nombre? Tampoco estamos por encima de la primitiva superstición de que podemos tener cierto control sobre algún demonio malicioso desconocido llamándolo por un nombre seguro, familiar e inocente, tal como "ingeniería". Pero esto es totalmente simbólico, así como demostró uno de los fabricantes de computadoras de los EE.UU. hace unos años cuando contrató, una noche, cientos de nuevos "ingenieros de software" mediante el simple mecanismo de elevar a todos sus programadores a ese exaltante rango. Demasiado para ese término.

La práctica está impregnada de la confortable ilusión de que los programas son simplemente dispositivos como cualquier otro, la única diferencia que se admite es que su fabricación pueden requerir un nuevo tipo de expertos, a saber: programadores. Desde allí hay sólo un pequeño paso hasta medir la "productividad del programador" en términos de la "cantidad de líneas producidas por mes". Esta es una unidad de medida muy costosa, porque anima a escribir código insípido, pero hoy estoy menos interesado en qué tan tonta es una unidad, aún desde un punto de vista puramente empresarial. Mi punto hoy es que, si deseamos contar líneas de código, no deberíamos verlas como "líneas producidas", sino como "líneas gastadas": el sentido común actual es tan tonto como contabilizar esa cuenta del lado erróneo del balance.

Además de la noción de productividad, también el control de calidad sigue estando distorsionado por la confortable ilusión de que funciona con otros aparatos como lo hace con los programas. Han pasado ya dos décadas desde que se señaló que el testing de programas puede convincentemente demostrar la presencia de errores, pero nunca puede demostrar su ausencia. Después de citar devotamente este comentario bien publicitado, el ingeniero de software vuelve al orden del día y continúa refinando sus estrategias de testing, tal como el alquimista de antaño, quien continuaba refinando sus purificaciones crisocósmicas.

Un profundo malentendido es luego revelado por el término "mantenimiento de software", como resultado del cual muchas personas siguen creyendo que los programas –e inclusive los mismísimos lenguajes de programación– están sujetos a desgaste y ruptura. Su auto también necesita mantenimiento, ¿no es así? Es famosa la historia de la empresa petrolera que creía que sus programas PASCAL no durarían tanto como sus programas FORTRAN "porque PASCAL no estaba mantenido".

En el mismo sentido debo llamar la atención sobre la sorprendente facilidad con que se ha aceptado la sugerencia de que los males de la producción de software de deben, en gran medida, a la falta de "herramientas de programación" apropiadas. (Pronto aparecería la frase "banco de trabajo del programador".) Nuevamente, la chatura de la analogía subyacente se debe a la Edad Media. Las confrontaciones con las insípidas "herramientas" del tipo de "animación de algoritmos" no ha suavizado mi juicio; por el contrario, ha confirmado mi sospecha inicial de que estamos tratando principalmente con otra dimensión del negocio del aceite de serpientes.

Finalmente, para corregir la posible impresión de que la inhabilidad de enfrentar la novedad radical está confinada al mundo industrial, déjenme ofrecerles una explicación de la –al menos americana– popularidad de la Inteligencia Artificial. Uno esperaría que la gente se sintiera aterrorizada por los "cerebros gigantes de máquinas que piensan". De hecho, la atemorizante computadora se vuelve menos atemorizante si es utilizada solamente para simular una no-computadora que nos es familiar. Estoy seguro de que esta explicación seguirá siendo controvertida por bastante tiempo, dado que la Inteligencia Artificial como imitadora de la mente humana prefiere verse a sí misma como a la vanguardia, mientras mi explicación la relega a la retaguardia. (El esfuerzo de utilizar máquinas para imitar la mente humana siempre me ha parecido bastante tonto: las usaría para imitar algo mejor.)

Hasta aquí la evidencia de que las novedades computacionales son, de hecho, radicales.

Y ahora viene la segunda –y más difícil– parte de mi charla: las consecuencias educativas y científicas de lo anterior. Las consecuencias educativas son, por supuesto, las más engorrosas, por lo tanto pospongamos su discusión y quedémonos mientras tanto con las ciencias de la computación en sí mismas. ¿Qué es la computación? ¿Y de qué se trata la ciencia de la computación?

Bien, una vez que todo está dicho y hecho, la única cosa que las computadoras pueden hacer por nosotros es manipular símbolos y producir resultados de tales manipulaciones. De nuestras observaciones previas, deberíamos recordar que este es un mundo discreto y, más aún, tanto los números como los símbolos involucrados así como la cantidad de manipulaciones realizadas son varios órdenes de magnitud mayores que los que podemos concebir: desconciertan totalmente nuestra imaginación y por lo tanto no debemos tratar de imaginárnoslos.

Pero antes de que una computadora esté lista para realizar alguna clase de manipulación con sentido –o cálculo, si se prefiere– debemos escribir un programa. ¿Qué es un programa? Varias respuestas son posibles. Podemos ver a un programa como lo que transforma una computadora de propósito general en un manipulador de símbolos de propósito específico, y lo hace sin necesidad de cambiar un solo cable (Esto fue una enorme mejora respecto de las máquinas con paneles de cables dependientes del problema.) Prefiero describirlo de la otra manera: un programa es un manipulador de símbolos abstracto, que puede convertirse en uno concreto suministrándole una computadora. Después de todo, el propósito de los programas ya no es más instruir a nuestras máquinas; en estos días, el propósito de las máquinas es ejecutar nuestros programas.

Por lo tanto, tenemos que diseñar manipuladores de símbolos abstractos. Todos sabemos cómo se ven: se ven como programas o –para usar una terminología más general– usualmente fórmulas de algún sistema formal un tanto elaboradas. Realmente ayuda ver a un programa como una fórmula. Primero, pone la tarea del programador en la perspectiva correcta: tiene que derivar esa fórmula. Segundo, explica por qué el mundo de las matemáticas ha ignorado el desafío de la programación: los programas eran fórmulas mucho más largas que las usuales, al punto que ni siquiera las reconocieron como tales. Ahora, de vuelta al trabajo del programador: tiene que derivar esa fórmula, tiene que derivar ese programa. Sabemos de una única forma confiable de hacerlo, mediante la manipulación de símbolos. Y ahora el círculo está cerrado: construimos nuestros manipuladores de símbolos mecánicos mediante la manipulación de símbolos humana.

Por lo tanto, la ciencia de la computación está –y siempre estará– relacionada con la interacción entre la manipulación de símbolos mecanizada y humana, usualmente llamadas "computación" y "programación", respectivamente. Un beneficio inmediato de esta visión es que revela a la "programación automática" como una contradicción en términos. Un beneficio posterior es que nos da una clara indicación acerca de dónde ubicar la ciencia de la computación en el mapa de las disciplinas intelectuales: en la dirección de la matemática formal y la lógica aplicada, pero finalmente mucho más allá de donde se encuentra actualmente, dado que la ciencia de la computación se interesa en el uso efectivo de los métodos formales en una escala mucho, mucho mayor de la que hemos sido testigos hasta ahora. Dado que ningún emprendimiento es respetable por estos días sin una STL (Sigla de Tres Letras) propongo que adoptemos para la ciencia de la computación IMF (Iniciativa de los Métodos Formales), y, para estar del lado seguro, mejor sigamos los brillantes ejemplos de nuestros líderes y hagamos de ella una Marca Registrada.

En el largo plazo espero que la ciencia de la computación trascienda a sus disciplinas padres, matemática y lógica, efectivamente realizando una parte significativa del Sueño de Leibniz de proveer un cálculo simbólico como una alternativa al razonamiento humano. (Por favor, note la diferencia entre "imitar" y "proveer una alternativa a": a las alternativas se les permite ser mejores.)

De más está decirlo, esta visión acerca de qué trata la ciencia de la computación no es universalmente aplaudida. Por el contrario, ha encontrado oposición –y a veces hasta violenta– desde todo tipo de direcciones. Menciono como ejemplos:

(0) la comunidad matemática, que quisiera continuar creyendo que el Sueño de Leibniz es una ilusión irreal

(1) la comunidad empresarial, quienes, habiéndoseles vendido la idea de que las computadoras harían la vida más simple, no están mentalmente preparados para aceptar que sólo resolvieron los problemas más simples al precio de crear uno mucho más difícil

(2) la subcultura del programador compulsivo, cuya ética prescribe que una idea tonta y un mes de codificación frenética deberían bastar para hacerlo millonario de por vida

(3) los ingenieros en computación, quienes quisieran continuar actuando como si fuera solamente cuestión de mayor flujo de bits o más flops por segundo

(4) la milicia, quienes están hoy totalmente absorbidos por el negocio de usar computadoras para transformar partidas de miles de millones de dólares en la ilusión de seguridad automática

(5) todo tipo de ciencias para las cuales la computación ahora actúa de alguna especie de refugio interdisciplinario

(6) el negocio educativo que siente que, si tiene que enseñar matemática formal a los estudiantes de Ciencias de la Computación, también debería cerrar sus escuelas.

Y con este sexto ejemplo he alcanzado, imperceptiblemente pero también inevitablemente, la parte más engorrosa de esta charla: consecuencias educativas.

El problema con la política educativa es que es difícilmente influenciada por consideraciones científicas derivadas de los tópicos dictados, y casi completamente determinada por circunstancias ajenas a la ciencia tales como las expectativas conjugadas de los estudiantes, sus padres y sus futuros empleadores, y el enfoque prevaleciente del rol de la universidad: el acento está en formar sus graduados para los trabajos de nivel inicial de hoy o en proveer a su alumnado con el bagaje intelectual y las actitudes que perduraran por otros 50 años? ¿Le damos rencorosamente a las ciencias abstractas solo un rincón lejano en el campus, o las reconocemos como el motor indispensable de la industria de alta tecnología? Aún si hacemos esto último, ¿reconocemos una industria de alta tecnología como tal si su tecnología pertenece principalmente a las matemáticas formales? ¿Proveen las universidades a la sociedad el liderazgo intelectual que necesita o sólo el entrenamiento que demanda?

La retórica académica tradicional está perfectamente dispuesta a dar a estas cuestiones las respuestas tranquilizadoras, pero no creo en ellas. A modo de ilustrar mis dudas, en un artículo reciente en "¿Quién gobierna Canadá?", David H. Flaherty groseramente establece que "Además, la élite de los negocios descarta a los académicos e intelectuales como ampliamente irrelevantes e impotentes."

Así, si miro en mi borrosa bola de cristal hacia el futuro de la educación en ciencias de la computación, veo sobrecogedoramente la deprimente imagen del "Negocio acostumbrado". A las universidades les seguirá faltando el coraje de enseñar ciencia dura, continuará orientando mal a los estudiantes, y cada

nuevo escalón de infantilización del currículum será exaltado como progreso educativo.

Hace un buen rato que tengo mi borrosa bola de cristal. Sus predicciones son invariablemente melancólicas y usualmente correctas, pero estoy bastante acostumbrado a eso y no me impiden darles unas pocas sugerencias, aún si es meramente un ejercicio vano cuyo único efecto es hacerlos sentir culpables.

Podemos, por ejemplo, comenzar limpiando nuestro lenguaje no denominando a un bug un bug, sino denominándolo un error. Es mucho mas honesto porque pone manifiestamente la culpa donde corresponde, es decir, en el programador que cometió el error. La metáfora animada del bug que se introdujo maliciosamente mientras el programador no estaba mirando es intelectualmente deshonesta ya que disfraza el hecho de que el error es propia creación del programador. Lo agradable de este simple cambio de vocabulario es que tiene un profundo efecto: mientras, antes, un programa con sólo un error solía ser "casi correcto", después de ello un programa con un error es simplemente "erróneo" (porque tiene un error).

Mi próxima sugerencia lingüística es más rigurosa. Se trata de confrontar el síndrome de "si-este-tipo-quiere-hablarle-a-ese-tipo": nunca se refieran a partes de programas o piezas de equipo en una terminología antropomórfica, ni permitan hacerlo a sus estudiantes. Esta mejora lingüística es mucho más difícil de implementar de lo que podrían pensar, y su departamento puede considerar la introducción de multas para las violaciones, digamos veinticinco centavos para estudiantes de grado, cincuenta centavos para estudiantes de postgrado y cinco dólares para miembros de la facultad: para final del primer semestre del nuevo régimen, habrán recolectado suficiente dinero para dos becas.

La razón para esta última sugerencia es que la metáfora antropomórfica –por cuya introducción podemos culpar a John von Neumann– es una enorme desventaja para cada comunidad informática que la ha adoptado. He encontrado programas que quieren cosas, saben cosas, esperan cosas, creen cosas, etc., y cada vez eso generaba confusiones evitables. La analogía que subyace a esta personificación es tan superficial que no es solamente engañosa sino también paralizante.

Es engañosa en el sentido que sugiere que podemos lidiar con el desconocido discreto en términos del familiar continuo, es decir, nosotros mismos, quod non. Es paralizante en el sentido que, debido a que las personas existen y actúan en el tiempo, su adopción efectivamente impide un despegue de la semántica operacional y fuerza así a la gente a pensar sobre los programas en términos de comportamientos computacionales, basados en un modelo computacional subyacente. Esto es malo, porque el razonamiento operacional es un tremendo desperdicio de esfuerzo mental.

Déjenme explicarles la naturaleza de ese tremendo desperdicio, y permítanme tratar de convencerlos de que el término "tremendo desperdicio de esfuerzo mental" no es una exageración. Por un breve lapso, me tornaré altamente técnico, pero no se acobarden: es el tipo de matemáticas que uno puede hacer con las manos en los bolsillos. El punto a comunicar es que si tenemos que demostrar algo respecto de todos los elementos de un conjunto grande, es desesperanzadamente ineficiente tratar con todos los elementos del conjunto individualmente: el argumento eficiente no se refiere a elementos individuales en lo absoluto y se lleva a cabo en términos de la definición del conjunto.

Consideren la figura plana Q, definida como el cuadrado de 8 por 8 del cual, en dos esquinas opuestas, han sido quitados dos cuadrados de 1 por 1. El área de Q es 62, que equivale al área combinada de 31 dominós de 1 por 2. El teorema es que la figura Q no puede ser cubierta por 31 de tales dominós.

Otra manera de exponer el teorema es que si comienza con papel cuadriculado y se cubre este ubicando cada siguiente dominó en dos nuevos recuadros adyacentes, ninguna distribución de 31 dominós dará como resultado la figura Q.

Así, una posible manera de probar el teorema es generando todas las posibles distribuciones de dominós y verificando para cada distribución que no da como resultado la figura Q.

El argumento simple, sin embargo, es como sigue. Pinte los recuadros del papel cuadriculado como un tablero de ajedrez. Cada dominó, cubriendo dos recuadros adyacentes, cubre 1 recuadro blanco y 1 negro, y , por consiguiente, cada distribución cubre tantos recuadros blancos como recuadros negros. En la figura Q, sin embargo, el número de recuadros blancos y el número de recuadros negros difiere en 2 –esquinas opuestas sobre la misma diagonal– y por consiguiente ninguna disposición de dominós da como resultado la figura Q.

No sólo es el simple argumento previo muchos órdenes de magnitud más corto que la investigación exhaustiva de las posibles distribuciones de 31 dominós, es también esencialmente más poderosa, dado que cubre la generalización de Q reemplazando el cuadrado original de 8 por 8 por cualquier rectángulo con lados de longitud par. Siendo infinito el número de tales rectángulos, el método previo de exploración exhaustiva es esencialmente inadecuada para probar nuestro teorema generalizado.

Y esto concluye mi ejemplo. Ha sido presentado porque ilustra de un tirón el poder de las matemáticas terrenales; no hace falta decirlo, la negación de explotar este poder de las matemáticas terrenales escala al suicidio intelectual y tecnológico. La moraleja de la historia es: tratar con todos los elementos de un conjunto ignorándolos y trabajando con la definición del conjunto.

Volvamos a la programación. La aseveración de que un programa dado cumple una cierta especificación escala a una aseveración sobre todos los cálculos computacionales que podrían ocurrir bajo el control de ese programa dado. Y dado que este conjunto de cálculos está definido por el programa dado, nuestra reciente moraleja dice: trate con todas los cálculos posibles bajo control de un programa dado ignorándolas y trabajando con el programa. Debemos aprender a trabajar con el texto de los programas mientras (temporalmente) se ignora que admiten la interpretación de código ejecutable.

Otra manera de decir la misma cosa es la siguiente. Un lenguaje de programación, con su sintaxis formal y las reglas de demostración que define su semántica, es un sistema formal para el cual la ejecución del programa provee solamente un modelo. Es bien conocido que los sistemas formales deberían ser tratados por derecho propio, y no en términos de un modelo específico. Y, de nuevo, el corolario es que deberíamos razonar sobre los programas sin siquiera mencionar su posible "comportamiento".

Y esto concluye mi excursión técnica en el motivo por el cual el razonamiento operacional sobre la programación es "un tremendo desperdicio de esfuerzo mental" y por qué, en consecuencia, en la ciencia de la computación debería prohibirse la metáfora antropomórfica.

No todo el mundo comprende esto suficientemente bien. Recientemente fui expuesto a una demostración de lo que pretendía ser software educativo para un curso introductorio de programación. Con sus "visualizaciones" en la pantalla era un caso tan obvio de infantilización del currículum que su autor debería ser acusado de su "menosprecio del cuerpo estudiantil", pero esto era sólo un daño menor comparado con para qué se usaban las visualizaciones: ¡se usaban para mostrar todo tipo de características de cálculos computacionales evolucionando bajo el control del programa del estudiante! El sistema

remarcaba precisamente aquello que el estudiante tiene que aprender a ignorar, reforzaba precisamente lo que el estudiante tiene que desaprender. Dado que quitarse los malos hábitos, más que adquirir nuevos, es la parte más dura del aprendizaje, debemos esperar de ese sistema un daño mental permanente para la mayoría de los estudiantes expuestos.

No hace falta decirlo, ese sistema ocultaba completamente el hecho de que, por sí sólo, un programa no es más que la mitad de una conjetura. La otra mitad de la conjetura es la especificación funcional que se supone que satisface el programa. La tarea del programador es presentar las conjeturas completas como teoremas demostrados.

Antes de irnos, me gustaría invitarlos a considerar la siguiente forma de hacer justicia a las novedades radicales de la computación en un curso introductorio a la programación.

Por un lado, enseñamos algo que se parece al cálculo de predicados, pero lo hacemos de manera muy distinta a los filósofos. A fin de entrenar al programador novato en la manipulación de fórmulas sin interpretar, lo enseñamos más como álgebra booleana, familiarizando al estudiante con todas las propiedades algebraicas de los conectivos lógicos. Para romper aún más los vínculos con la intuición, renombramos los valores {verdadero, falso} del dominio booleano como {negro, blanco}

Por otro lado, enseñamos un lenguaje de programación imperativo simple y claro, con un skip y una asignación múltiple como sentencias básicas, con una estructura de bloque para variables locales, el punto y coma como operador para composición de sentencias, una bonita construcción alternativa, una bonita repetición y, si se quiere, una llamada a procedimiento. A esto agregamos un mínimo de tipos de datos, digamos booleanos, enteros, caracteres y cadenas. Lo esencial es que, para lo que sea que introduzcamos, la semántica correspondiente está definida por las reglas de demostración que la acompañan.

Desde el comienzo, y a través de todo el curso, enfatizamos que la tarea del programador no es sólo escribir un programa, sino que su tarea principal es dar una prueba formal de que el programa que propone cumple la especificación funcional (igualmente formal). Mientras se diseñan demostraciones y programas conjuntamente, el estudiante adquiere una amplia oportunidad de perfeccionar su destreza manipulativa con el cálculo de predicados. Finalmente, para hacer llegar el mensaje de que este curso introductorio a la programación es principalmente un curso en matemáticas formales, nos encargamos de que el lenguaje de programación en cuestión no haya sido implementado en el campus de manera que los estudiantes estén protegidos de la tentación de probar sus programas. Y esto concluye el esbozo de mi propuesta para un curso introductorio a la programación para estudiantes de primer año.

Esta es una propuesta seria, y sumamente sensible. Su única desventaja es que es demasiado radical para muchos, quienes, siendo incapaces de aceptarla, se ven forzados a inventar alguna justificación rápida para desestimarla, no importa cuan inválida sea. Les daré unas pocas de esas justificaciones.

No necesitan tomar mi propuesta seriamente porque es tan ridícula que estoy obviamente desconectado del mundo real. Pero ese barrilete no va a volar, ya que conozco el mundo real demasiado bien: los problemas del mundo real son principalmente aquellos con los que ustedes se quedan después de negarse a aplicar sus efectivas soluciones. Así que, probemos de nuevo.

No necesitan tomar seriamente mi propuesta porque es sumamente surrealista intentar enseñar tal material a alumnos de primer año. ¿No sería esa una salida fácil? Acaban de postular que esto sería por lejos muy difícil. Pero ese barrilete tampoco va a volar, dado que el postulado se ha demostrado falso: desde

principios de los ’80, se ha dado tal curso introductorio a la programación a cientos de estudiantes de primer curso de grado cada año. [Porque, en mi experiencia, decir esto no es suficiente, la sentencia previa debería repetirse al menos otras dos veces]. Así que, intentemos de nuevo.

Admitiendo renuentemente que podría quizás enseñarse a estudiantes suficientemente dóciles, todavía rechazan mi propuesta porque tal curso se desviaría tanto de lo que los estudiantes de 18 años están habituados y esperan, que inflingírselos sería un acto de irresponsabilidad educativa: sólo frustraría a los estudiantes. No hace falta decirlo, ese barrilete tampoco va a volar. Es cierto que el estudiante que nunca ha manipulado fórmulas sin interpretar se da rápidamente cuenta que se confronta con algo totalmente distinto a cualquier cosa que haya visto antes. Pero afortunadamente, las reglas de manipulación son en este caso tan pocas y simples que muy poco después hace el excitante descubrimiento de que está comenzando a dominar el uso de una herramienta que, en toda su simplicidad, le da un poder que sobrepasa sus sueños más audaces.

Enseñar a jóvenes desprevenidos el uso efectivo de los métodos formales es uno de los placeres de la vida porque es extremadamente gratificante. En pocos meses, encuentran su camino en un mundo nuevo con justificado grado de confianza, que es radicalmente novedoso para ellos; en pocos meses, su concepto de cultura intelectual ha adquirido una dimensión radicalmente novedosa. Para mi gusto y estilo, esto es de lo que se trata la educación. Las universidades no deberían temer a enseñar novedades radicales; por el contrario, es su llamado dar la bienvenida a la oportunidad de hacerlo. Su disposición a hacerlo es nuestra principal salvaguarda contra las dictaduras, sean del proletariado, del establishment académico, o de la élite corporativa.

Austin, 2 de Diciembre de 1988

prof. dr. Edsger W. Dijkstra

Departamento de Ciencias de la Computación

Universidad de Texas

Austin, TX 78712-1188

USA

Traducción: Javier Smaldone y Billy Biset.

Transcripción original: Javier Smaldone.

Última revisión: 5 de agosto de 2006 (11:45 am).

Última versión y actualizaciones: http://www.smaldone.com.ar/documentos/ewd.shtml

Nota de los traductores: La versión HTML contiene el texto original en forma de comentarios, para simplificar su corrección.