Some Comments on the Aims of Mirfac (EWD-68)

Edsger Wybe Dijkstra

Algunos Comentarios a los Objetivos del Mirfac

Traducción

Edgar Serna M.

Fundación Universitaria Luis Amigó

edgar.sernamo@amigo.edu.co

Nota del traductor. En 1963, H. James Gawlik publicó su artículo "MIRFAC: A Compiler Base on Standard Mathematical Notation and Plain English", en el que describe una versión piloto del compilador MIRFAC. Las principales características del sistema, de acuerdo con el autor, es que está destinado a la solución de problemas científicos con presentación totalmente de fórmulas matemáticas. El uso del inglés sin formato para las instrucciones organizacionales, diagnostica los errores de manera automática e indica la ubicación real del error en un programa sin compilar. Además es un intento por minimizar la fragmentación de la declaración del problema original, que es una característica normal de los sistemas de programación. En esta carta, Dijkstra le escribe al editor de la revista, Communications of the ACM, sus comentarios acerca de dicho artículo.

Estimado Editor,

Recientemente H. J. Gawlik [1] ha publicado un artículo acerca del proyecto MIRFAC: A Compiler Base on Standard Mathematical Notation and Plain English. Su autor está al corriente de proyectos anteriores en líneas análogas (MADCAP y MADCAP) [2]. Cuando me enteré de esos proyectos anteriores me asombré debido a lo que pretendían parecer para mí difícilmente era algo sensato. No levanté mi voz entonces, convencido y confiado en que las personas descubrirían esto por sí mismas en un tiempo muy corto. Ahora, dos años y medio después me enfrento con el hecho de que el movimiento no ha desaparecido por muerte natural, como yo había supuesto que iba a suceder. Este descubrimiento me ha causado algo de decepción y sólo puedo lamentar mi silencio anterior sobre el tema.

La justificación para el proyecto MIRFAC parece ser la opinión de que lo que es correcto para la comunicación de persona a persona también debe ser adecuado para la comunicación del hombre con la máquina. (Esto es la única interpretación que me permite darle un significado a la declaración de Gawlik de que "un compilador no debe tener como objetivo simplemente simplificar la programación, sino también abolirla"). ¡Pero esta opinión no debería pasar sin respuesta!

Si encargamos a una persona "inteligente" de hacer algo por nosotros, podemos caer nosotros mismos en todo tipo de descuidos, errores, faltas, contradicciones, etc., apelando a su comprensión y sentido común: no se espera que lleve a cabo literalmente las tonterías que se le indicaron, se espera que intente hacer lo que se ordenó. Y un empleado humano es por lo tanto útil en virtud de su "desobediencia". Esto puede ser de alguna conveniencia para el maestro que no le gusta expresarse con claridad; el precio que se paga es el no depreciable riesgo de que el empleado realice, por su propia cuenta, algo totalmente involuntario.

Si, sin embargo, instruimos a una máquina para hacer algo debemos ser conscientes del hecho de que por primera vez en la historia de la humanidad tenemos un empleado a nuestra disposición que realmente hace lo que se ha dicho que haga. En la comunicación hombre-máquina no sólo es necesario ser excepcionalmente precisos y claros, hay por fin, además un punto para serlo, si deseamos por lo menos obtener todos los beneficios del poderoso y obediente empleado mecánico. Los esfuerzos encaminados a ocultar esta nueva necesidad de precisión —para el supuesto beneficio del usuario—, de hecho son perjudiciales: porque al mismo tiempo desean ocultar las igualmente nuevas posibilidades de la computación automática, de tener los procesos complejos bajo completo control.

Sigo citando al Sr. Gawlik: "... MIRFAC ha sido desarrollado para satisfacer el criterio básico de que sus enunciados del problema deben ser comprensibles para los no programadores, con el doble objetivo de que el usuario no necesita aprender algún lenguaje que no conozca ya, y que la exactitud del enunciado del problema pueda ser comprobada por cualquiera que entienda el problema aunque no conozca nada de programación".

No veo el punto del señor Gawlik acerca del "criterio fundamental". En otras partes he sido advertido acerca de la "... tendencia por diseñar lenguajes de programación para que puedan ser leídos fácilmente por un lector semi-profesionales, semi-interesado" (ver [3]). (Los síntomas de esta tendencia son lenguajes suyo vocabulario incluye una agreste variedad de palabras en Inglés que se utiliza en un sentido casi normal, y algunos traductores que incluso permiten una lista en constante expansión de sinónimos y faltas de ortografía para esas palabras. Particularmente, los lenguajes diseñados bajo la presión comercial han sufrido seriamente de esta tendencia). Se ve tan atractiva: "Todo el mundo puede comprenderlo inmediatamente". Pero dar una interpretación semántica plausible para un texto que uno supone que es correcto y significativo, es una

cosa, escribir tal texto [.....] expresando exactamente lo que se quiere decir, "¡puede ser un asunto muy diferente"! Por razones similares, John McCarthy dice que "COBOL... es un paso hacia un callejón sin salida debido a su orientación hacia el Inglés, que no es adecuado para describir formalmente los procedimientos" [4].

Por otra parte, lograr el doble objetivo del Sr. Gawlik es un auto engaño. La notación matemática estándar ha sido diseñada para describir relaciones y ahora tenemos que definir los procesos. El inglés sin formato ha surgido de una necesidad de comunicación interhumana, pare se vago y ambiguo, para contar chistes y cantar canciones de cuna, pero obviamente es inútil para expresar lo que tiene que expresarse ahora. Uno puede apropiarse de notaciones matemáticas, puede pedir prestadas palabras en inglés, pero una semántica completamente nueva se debe adjuntar a ellos y a pesar de una semejanza superficial uno crea un nuevo lenguaje. Y creo que la similitud es más engañosa que clarificadora.

Este temor se ve confirmado por el segundo objetivo del Sr. Gawlik, a saber: "que la exactitud del enunciado del problema pueda ser comprobada por cualquiera que entienda el problema aunque no conozca nada de programación". Por supuesto que puede comprobarla, pero el punto crucial es si ¡va a encontrar los errores! Y por supuesto que no los encontrará: porque en la comunicación humana uno es constantemente entrenado para tratar de comprender las intenciones de los demás sin notar las tonterías. El corrector que entiende el problema pero que no conoce nada de programación, será engañado por la familiaridad de los caracteres y las palabras y, con toda probabilidad, estará satisfecho si reconoce el problema.

Estoy totalmente a favor de los lenguajes algorítmicos claros y convenientes, pero, por favor, francamente dejar que: disimulen en apariencia que han sido adaptados a otros propósitos, puede solamente incrementar la confusión.

E. W. Dijkstra
Department of Mathematics
Technological University
Postbox 513
EINDHOVEN
The Netherlands

REFERENCIAS

[1] H. J. Gawlik. "MIRFAC: A Compiler Based on Standard Mathematical Notation and Plain English". *Comm. ACM*, Vol. 6, No. 99, Sep. 1963.
[2] M. B. Wells. "MADCAP: A Scientific Compiler is a Displayed Formula Textbook Language". *Comm. ACM*, Vol. 4, No. 1, Jan. 1961.
[3] E. W. Dijkstra. "On the Design of Machine Independent Programming Languages". In Goodman, R. (Ed) "Annual Review in Automatic Programming", Vol. III. London: Pergamon Press, 1961.
[4] J. McCarthy. "A Basis for a Mathematical Theory of Computation, Preliminary Report". *Western Joist Computer Conference*. Los Angeles, California, May 09-11, 1961. Ω