Propuesta de Protocolo de Formación de Pares Experimentales de Programadores

Mauricio Dávila[1], Marisa Panizzi[1], Darío Rodríguez[2,3]

[1] Programa de Maestría en Ingeniería de Sistemas de Información. Universidad Tecnológica Nacional. Facultad Regional Buenos Aires, Castro Barros 91, (C1178AAA), C.A.B.A, Argentina.
[2] Grupo de Ingeniería de Espacios Virtuales de Trabajo y Grupo de Investigación en Sistemas de Información. Departamento de Desarrollo Productivo Tecnológico. Universidad Nacional de Lanús, 29 de Septiembre 3901, (B1826GLC), Lanús, Buenos Aires, Argentina.
[3] Comisión de Investigación Científicas - CIC. Calle 526 e/10 y 11, (B1906APP), La Plata, Buenos Aires, Argentina.
davilamr.80@gmail.com, marisapanizzi@outlook.com, dariorodriguez1977@gmail.com

Resumen. En este trabajo, se propone un protocolo que permite formar pares experimentales homogéneos de programadores con el propósito de asegurar que dos sujetos de iguales características son indistinguibles en lo que refiere a sus aptitudes como programadores. Este protocolo permite evaluar las características de los lenguajes de programación sin que estas mediciones se vean afectadas por las habilidades o ausencias de las mismas por parte de los programadores. Se presenta un caso testigo que permita validar el protocolo, a tal efecto se aplicará dicho protocolo a un grupo de programadores del lenguaje C con el fin de demostrar que los pares experimentales formados no presentan diferencias significativas, en lo que refiere a calidad y tiempo, en la codificación de una especificación.

Palabras Clave: experimentación en ingeniería de software, protocolo, pares experimentales de programadores, programación

1 Introducción

En la actualidad existe una gran diversidad de lenguajes de programación y a menudo esto dificulta la tarea de seleccionar el lenguaje que mejor se adapta a las necesidades del desarrollo. La decisión del lenguaje de programación para realizar una solución implica múltiples factores de análisis, muchos de los cuales están sujetos al código fuente producido y al tiempo empleado para poder producirlo.

Utilizando el método de revisiones sistemáticas [1] se ha realizado una investigación documental de estudios en los cuales se hace referencia a las métricas que se pueden establecer sobre el código producido con un lenguaje de programación determinado [2][3][4] y [5]. Cuando se utilizan métricas para determinar si un lenguaje es una mejor alternativa frente a otro, no se debe perder de vista que dichas métricas solo se enfocan en el código resultante y no tienen en consideración las características

del programador que construyó el código. Esto puede ocasionar la sub calificación o sobre calificación de un lenguaje por habilidades o ausencias de las mismas por parte de los programadores que los utilizan. Esto último, plantea un problema a la hora de diseñar un experimento cuyo propósito sea determinar el lenguaje a utilizar, ya que el grupo de programadores que utilizan el lenguaje A puede poseer un nivel de conocimientos superior que el grupo de programadores que emplean el lenguaje B o viceversa y esto impactaría en el resultado del experimento. Una posible solución a esta problemática sería contar con un conjunto de programadores que sean capaces de resolver la misma tarea en los lenguajes que se requieren evaluar.

Más allá de la dificultad que sería reunir a este conjunto de sujetos con los conocimientos necesarios en cada uno de los lenguajes que se pretenden evaluar, esta solución presenta algunas dificultades adicionales que surgen a la hora de diseñar experimentos.

Tanto en el trabajo de Juristo y Moreno [6] como en el de Wohlin [7] se considera que los experimentos realizados en el campo de la ingeniería de software son fuertemente influenciados por las características de los sujetos al igual que ocurre en otras ciencias, generalmente conocidas como ciencias sociales. Juristo y Moreno [6] enumeran algunos puntos relacionados con los factores sociales y las características específicas de desarrollo de software que deben tenerse en cuenta al diseñar experimentos.

- Efecto aprendizaje: Si un sujeto tiene que resolver el mismo problema aplicando diferentes lenguajes de programación es muy probable que este aprenda cada vez más sobre el problema y que el último resultado sea mejor que el primero, simplemente porque el sujeto sabe más sobre el problema y no porque el lenguaje de programación sea mejor.
- Efecto de aburrimiento: los sujetos se aburren o se cansan del experimento y ponen menos esfuerzo e interés a medida que pasa el tiempo.
- Efecto de entusiasmo: Puede suceder que los sujetos que utilizan un lenguaje de programación antiguo no están motivados para hacer un buen trabajo, mientras que los que utilizan un nuevo lenguaje de programación si lo están.
- Efecto de la experiencia: A la hora de realizar un experimento que involucra programadores es de esperar que existan distintos niveles tanto de conocimiento como de habilidad sobre el lenguaje de programación empleado.
- La formalización inconsciente: Surge cuando un mismo sujeto utiliza dos o más lenguajes de programación con diferentes grados de definición o formalidad.
- Efecto de ajuste: El estado emocional de los sujetos participantes se encuentra íntimamente relacionado con el rendimiento que estos tendrán.

A continuación se describen un conjunto de acciones a considerar para controlar los efectos antes enumerados:

- Efecto Aprendizaje: No utilizar al mismo conjunto de sujetos para desarrollar en más de un lenguaje de programación.
- Efecto de aburrimiento: Motivar a los sujetos que ejecutan el experimento de igual manera independientemente del grupo que integren.

- Efecto de entusiasmo: No poner al tanto a los sujetos sobre las hipótesis formuladas o los objetivos del experimento.
- Efecto de la experiencia: Para controlar este efecto se formarán pares experimentales de programadores que sean indistinguibles en lo que refiere a sus conocimientos y habilidades con el lenguaje de programación.
- La formalización inconsciente: Deberán ser considerados aspectos referidos al nivel de conocimiento que cada sujeto tiene en lo que refiere al lenguaje de programación empleado en el momento de formar el par experimental
- Efecto de ajuste: Se debe de tener en cuenta que todas las instancias del experimento se realicen bajo las mismas condiciones.

Dentro del área de ingeniería de software, es habitual enfrentarse a la necesidad de realizar experimentos con un conjunto reducido de personas las cuales aplican diferentes tratamientos a los objetos de estudio. Teniendo en cuenta que realizar comparaciones dentro de pares homogéneos de unidades experimentales no solo aumenta la precisión del análisis sino que también permite controlar gran parte de los efectos no deseados [6] a la hora de realizar un experimento. Este trabajo se desarrolla bajo la hipótesis de que es posible formar pares experimentales homogéneos de programadores y que por lo tanto dos sujetos de iguales características son indistinguibles en lo que refiere a sus aptitudes como programadores. De esta hipótesis se deriva las siguientes preguntas de investigación: ¿es posible formar pares experimentales indistinguibles en lo que refiere a sus aptitudes como programadores? De ser posible formarlos, ¿requieren el mismo tiempo para resolver con el mismo lenguaje la misma tarea?.

En la (Sección 2) se propone el protocolo de formación de pares experimentales de programadores, en la (Sección 3) se lleva adelante la validación del protocolo mediante la realización de una prueba piloto y en la (Sección 4) se plantean las conclusiones y las futuras líneas de investigación.

2 Propuesta del protocolo de formación de pares experimentales.

A la hora de definir cómo formar pares homogéneos de programadores, los autores adherimos a la propuesta de Campbell [8], que sostiene que muchos son los factores afectan el rendimiento de un individuo de manera indirecta, pero sólo hay tres determinantes directos del rendimiento: *conocimiento, habilidad y motivación*, por lo cual, se elaborará un protocolo que tenga por propósito la formación de pares experimentales de programadores que resulten homogéneos en lo que refiere tanto a nivel de conocimiento como a sus habilidades y dará por sentado que todos los participantes a caracterizar no presentan diferencias significativas en lo que respecta a la motivación.

El primero de los pasos consiste en identificar las formas utilizadas para categorizar a los programadores en otros estudios de investigación experimental (Sección 2.1), luego se describen los lineamientos considerados para la construcción de los instrumentos de caracterización utilizados en el experimento (Sección 2.2) y por último

en la (Sección 2.3) se establece un mecanismo que permite formar pares experimentales homogéneos de programadores en base a la caracterización realizada.

2.1 Experiencia de los programadores

Al igual que en la mayoría de las actividades humanas, el rendimiento individual en el desarrollo de software varía considerablemente de una persona a otra y se deben articular mecanismos para que dichas variaciones no afecten los resultados del estudio. Feigenspan y otros [9] realizaron un trabajo documental en el cual se analizaron 161 publicaciones y se identificaron nueve formas utilizadas por los investigadores para determinar la experiencia que un programador posee. Estos autores definen la experiencia como la cantidad de conocimientos adquiridos respecto al desarrollo de programas.

- Años: En cuarenta y siete de los trabajos, los años que un participante ha estado programando en general o en una empresa o en cierto lenguaje se usó para determinar la experiencia de programación.
- Educación: La educación de los participantes se utilizó para indicar su experiencia en diecinueve de los artículos revisados, en los cuales se incluyó información sobre el nivel de educación alcanzado (pregrado, grado, posgrado, etc.) o las calificaciones de los cursos.
- Auto Estimación: En doce trabajos, se pidió a los participantes que estimaran su propia experiencia.
- Cuestionario específico: En nueve trabajos los autores aplicaron un cuestionario para evaluar la experiencia de programación.
- Tamaño: El tamaño de los programas que los participantes habían escrito fué utilizado como indicador en seis artículos.
- Examen: En tres trabajos se realizó un examen de programación para evaluar la experiencia de los participantes.
- Supervisor: En dos trabajos, en los que los programadores profesionales fueron utilizados como participantes, un supervisor fue el encargado de estimar su experiencia.
- No especificado: A menudo, los autores afirman que midieron la experiencia de programación, pero no especificaron cómo. Este fue el caso en treinta y nueve trabajos.
- No controlada: La experiencia de programación no se mencionó en absoluto en cuarenta y cinco trabajos, lo que amenaza la validez de los experimentos correspondientes.

2.2 Caracterización de los programadores

Se requiere desarrollar un método de caracterización que no se base en la percepción que tiene cada individuo acerca de sus habilidades como programador ya que las personas menos competentes tienden a sobrestimar su habilidad al no tener conocimiento suficiente para reconocer sus propias limitaciones y que también es común que personas más preparadas tiendan a subestimar sus logros y sus competencias

[12]. Esta caracterización persigue como único objetivo calificar un conjunto de aptitudes del programador a solo efecto de poder encontrar parejas de programadores que se puedan considerar homogéneas. El propósito que tiene la caracterización es asegurar que dos sujetos de iguales características son indistinguibles en lo que refiere a sus aptitudes como programadores, para lo cual, se optó por analizar un amplio conjunto de aptitudes del programador, los autores adhieren a la idea que la realización de una caracterización basada tan solo en unos pocos criterios puede incurrir en graves errores. Algunas características de los programadores son independientes del lenguaje de programación y otras son dependientes de él, por lo cual se fijarán los lineamientos que permitan desarrollar dos instrumentos de caracterización, uno independiente del lenguaje de programación (CILP) y el otro que tenga en cuenta un lenguaje de programación (CDLP).

Existe un gran número de medidas para capturar atributos de los procesos y productos software, que tradicionalmente se han realizado confiando en la sabiduría de los expertos y esta situación ha conducido frecuentemente a cierto grado de imprecisión en las definiciones, propiedades y suposiciones de las medidas, haciendo que el uso de las medidas sea difícil, la interpretación peligrosa y los resultados de muchos estudios de validación contradictorios [10]. Para la construcción de las herramientas de caracterización se ha tenido en cuenta el procedimiento general de construcción de un instrumento de medición propuesto por Sampiere [11] y el método para la definición de medidas válidas propuesto por Genero, Cruz-Lemus y Piattini [10], adaptando dichos procedimientos a la necesidad de este trabajo.

Por cuestiones relacionadas a la síntesis que esta publicación demanda, no es posible detallar cada uno de los pasos que han permitido definir los instrumentos ni el contenido de cada dimensión de los mismos.

En la Tabla 1, se presentan los dominios del contenido de la variable (dimensiones), los indicadores de cada dimensión, y la nomenclatura propuesta de las dimensiones para el instrumento que se empleará para la caracterización independiente del lenguaje de programación.

En la Tabla 2, se presentan los dominios del contenido de la variable (dimensiones), los indicadores de cada dimensión, y la nomenclatura propuesta de las dimensiones para el instrumento que se empleará para la caracterización dependiente del lenguaje de programación.

En lo que respecta a la decisión del tipo y formato del instrumento y su contexto de administración se utilizará el procedimiento de recolección de datos mixto conformado por dos cuestionarios (CILP y CDLP) y luego por una entrevista.

Con el propósito de minimizar los errores de caracterización, una vez que el participante ha contestado el cuestionario se desarrollara una entrevista individual donde el entrevistador realiza preguntas a efectos de que el participante justifique sus respuestas y ante la correcta justificación del participante el entrevistador dará la misma como válida.

El contexto de administración será una sala provista de una computadora por programador ya que la primer instancia, la que refiere al cuestionario, es auto administrada permitiendo llevar esta acción de forma individual o grupal, para luego si pasar a la instancia de entrevista de forma individual.

Tabla 1. Variable, dimensiones, nomenclatura de cada dimensión y sus indicadores.

Variable a medir	Dimensión	Nomenclatura	Indicador
Características independientes del lenguaje de programación	*Nivel de formación.*	***CILP1***	Refiere al nivel alcanzado por el participante en su educación ya sea formal o informal.
	Experiencia	***CILP2***	Refiere a los años de experiencia como programador y a la cantidad de lenguajes que manifiesta conocer.
	Comprensión de una Especificación	***CILP3***	Capacidad que tiene el programador para comprender una especificación simple y el tiempo insumido.
	Comprensión de Pseudocódigo	***CILP4***	Capacidad que tiene el programador de interpretar el funcionamiento de bloques de pseudocódigo y el tiempo insumido.
	Capacidad Algorítmica	***CILP5***	Capacidad que tiene el programador en desarrollar una solución con pseudocódigo y el tiempo que emplea en realizar dicha solución.

Tabla 2. Variable, dimensiones, nomenclatura de cada dimensión y sus indicadores.

Variable a medir	Dimensión	Nomenclatura de la Dimensión	Indicador
Características dependientes del lenguaje de programación	*Lenguaje Elegido*	***CDLP1***	Refiere al lenguaje de programación que el participante expresa dominar con mayor fluidez y a los años de experiencia que tiene con este.
	Conocimiento Teórico	***CDLP2***	Refiere al nivel de conocimientos sobre aspectos teóricos del lenguaje de programación que el participante posee.
	Comprensión de Código Fuente	***CDLP3***	Capacidad que tiene el programador de interpretar el funcionamiento de bloques de código y el tiempo insumido en ello.

2.3 Mecanismo de formación de pares experimentales

Es pertinente contar con un mecanismo que asegure que las parejas experimentales se encuentran integradas por sujetos homogéneos, esto implica que según su caracterización son indistinguibles o presentan diferencias mínimas.

2.3.1 Criterio de Utilización de las Variables

Del procedimiento de caracterización surgen múltiples variables, algunas relacionadas a características independientes del lenguaje de programación y otras a características que dependen del desempeño del programador en un lenguaje de programación en particular.

2.3.2 Normalización

El proceso de normalización implica transformar los valores de las variables independientes a fin de que estas queden representadas en el rango [0-10] independientemente de cual fuese su escala original. Este paso, permitirá asegurar que ninguna de las variables intervinientes en el cálculo de distancia se encuentran ponderadas por sobre otras.

2.3.3 Penalidades por tiempo

Cada una de las variables medida es acompañada por el tiempo insumido por el participante en completar los ejercicios relacionados a la misma. A efectos de formar pares experimentales que no solo sean indistinguibles en lo que refiere a conocimiento sino también al tiempo insumido para dar dicha respuesta, se aplicará una penalidad en el puntaje obtenido según el tiempo insumido, la puntuación obtenida se irá decrementando en un 10% por cada cinco minutos insumidos.

2.3.4 Cálculo de Distancia

En un escenario donde se requiere evaluar múltiples variables, todas ellas cuantitativas y cuyos valores pertenecen luego del proceso de normalización al intervalo [0,10] se debe definir el criterio a ser empleado para determinar la distancia existente entre dos sujetos, por tratarse de un espacio n-dimensional se aplicará el cálculo de distancia euclídea [15] (Figura 1).

$$d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_i - q_i)^2 + \cdots + (p_n - q_n)^2}.$$

Figura 1. Distancia euclídea [15]

Se realizará el cálculo de distancia entre participantes donde el valor del módulo de la diferencia entre variables del mismo tipo no supere un umbral, esta restricción permitirá establecer una distancia máxima tolerada la cual no podrá ser producto de una diferencia significativa en una sola variable.

2.3.4 Algoritmo de Selección de Parejas Experimentales

El algoritmo sigue un proceso secuencial en el que a cada paso toma una decisión, seleccionar la mínima distancia de entre las disponibles y luego en el siguiente paso el algoritmo se encuentra con un problema idéntico, pero estrictamente menor, al que tenía en el paso anterior y vuelve a aplicar la misma función de selección para tomar la siguiente decisión [14].

3 Caso de validación del protocolo.

Se realizaron una serie de acciones para demostrar el nivel de confiabilidad y validez inicial de los instrumentos de medición. Las características que intervienen en la formación de los pares experimentales son: Comprensión de una Especificación (CILP3), Comprensión de Pseudocódigo (CILP4), Capacidad Algorítmica (CILP5), Conocimiento Teórico (CDLP2) y Comprensión de Código Fuente (CDLP3).

Para la realización de la prueba piloto inicial, se convocó a participar de la investigación a personas mayores de edad que han manifestado conocer el lenguaje de programación C, logrando conformar un grupo de 14 programadores. Luego de ser caracterizados los participantes, se les solicitó resolver una tarea de mediana complejidad, por último se aplicó el protocolo de formación de pares experimentales de programadores a efecto de poder determinar si cada pareja experimental presenta o no diferencias significativas en la resolución de la tarea. Si los pares experimentales conformados aplicando el protocolo no presentan diferencias significativas al resolver con el mismo lenguaje la misma tarea, se podrá considerar que tanto los instrumentos como el protocolo de formación de pares experimentales se encuentran en una versión estable.

Tabla 3. Resultados normalizados de la caracterización.

Sujeto	CDLP2	CDLP3	CILP3	CILP4	CILP5
1	2.40	0.00	10.00	0.00	1.64
2	8.10	6.75	9.00	5.00	6.36
3	6.40	4.00	7.50	5.00	2.73
4	9.00	9.00	4.50	5.00	6.36
5	6.30	2.25	6.75	7.00	2.18
6	7.20	6.00	4.50	9.00	0.67
7	8.10	0.00	7.50	4.50	1.48
8	5.60	2.25	10.00	4.50	4.43
9	8.00	4.50	6.75	9.00	2.96
10	6.30	4.50	4.50	4.50	7.64
11	7.20	2.50	6.75	5.00	1.86
12	9.00	6.75	9.00	9.00	5.45
13	5.60	0.00	7.50	5.00	7.00
14	6.40	0.00	9.00	4.50	3.87

Los resultados que surgen del proceso de caracterización figuran en la (Tabla 3). En la (Tabla 4) se presenta la matriz de distancia correspondiente, las intersecciones pintadas de color negro representan pares de sujetos que no deben ser considerados ya que en por lo menos una de sus dimensiones han presentado una distancia superior al cincuenta por ciento. Los pares experimentales que surgen de aplicar el algoritmo se encuentran resaltados en color gris. Por último se presentan las diferencias de tiempo que existen entre los programadores que integran cada par experimental (Tabla 5).

Tabla 4. Matriz de distancia.

ID	2	4	5	7	9	11	12	14
2	-	5.11	7.07		6.14	6.65	4.20	
4	5.11	-			7.34		6.49	
5	7.07		-	3.95	3.54	2.23	6.88	4.38
7			3.95	-	6.58	2.83		9.20
9	6.14	7.34	3.54	6.58	-	4.67	4.17	7.00
11	6.65		2.23	2.83	4.67	-	7.44	4.03
12	4.20	6.49	6.88		4.17	7.44	-	
14			4.38	9.20	7.00	4.03		-

Tabla 5. Diferencias de tiempo empleado entre integrantes del par experimental

Sujetos		Diferencia	
		Tiempo	%
2	4	6	7.06%
5	11	3	2.48%
7	14	5	4.76%
9	12	5	4.35%

En lo referente al tiempo empleado para realizar la caracterización, el tiempo promedio utilizado para llevar adelante la caracterización independiente del lenguaje de programación fue de 28 minutos, para la caracterización dependiente del lenguaje de programación el tiempo promedio empleado fue de 19 minutos y para llevar a cabo las entrevistas en promedio se utilizaron 6 minutos por participante. Puede observarse en la (Tabla 5) que los pares experimentales de programadores no presentan diferencias significativas en el tiempo empleado para resolver la misma tarea.

4 Conclusiones y Futuros Trabajos.

Con el fin de proponer un protocolo de formación de pares experimentales de programadores se ha llevado a cabo un análisis documental en lo que refiere a los beneficios que esta forma de experimentación aporta en el área de ingeniería de software. Se ha logrado la construcción de dos instrumentos que permiten caracterizar a los programadores. Se ha definido el procedimiento que permite formar los pares experimentales a partir de los datos obtenidos con los instrumentos de caracterización.

Por último se ha verificado mediante un caso de validación si los pares experimentales obtenidos presentaban diferencias en lo que refiere al tiempo empleado en resolver una misma tarea de programación.

Se puede concluir que en lo que refiere a la formación de los pares experimentales de programadores, el protocolo ha funcionado satisfactoriamente y por lo tanto se considera que tiene un nivel de confiabilidad, validez y objetividad aceptable ya que demostró consistencia entre los resultados obtenidos.

Como trabajo futuro se identifica la necesidad de: (1) poder aplicar el protocolo en otros lenguajes de programación, (2) poder utilizar el protocolo para formar pares experimentales entre sujetos que utilizan distintos lenguajes de programación a efectos de poder medir si un lenguaje de programación en particular afecta o no a la productividad informática.

Referencias

1. Argimón, J. Métodos de Investigación Clínica y Epidemiológica. Elsevier España. 84-8174-709-2. (2004).
2. Halstead, M.. Elements of Software Science. Elsevier Science Inc., New York, NY, USA.(1977).
3. McCabe, T. J. A complexity measure. IEEE Transactions on software Engineering , 4, 308-320. 1976.
4. Riaz, M., Mendes, E., & Tempero, E. A systematic review of software maintainability prediction and metrics. In Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement (pp. 367-377). IEEE Computer Society. 2009.
5. Rilling, J., y Klemola, T. Identifying Comprehension Bottlenecks Using Program Slicing and Cognitive Complexity Metrics. 10th IEEE Working Conference on Reverse Engineering. 10, pp. 115-125. Oregon, USA: IEEE. 2003.
6. Juristo, N., & Moreno, A. M. Basics of software engineering experimentation. Springer Science & Business Media. 2013
7. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., & Wesslén, A. Experimentation in software engineering. Springer Science & Business Media. 2012.
8. Campbell, J. P., McCloy, R. A., Oppler, S. H., & Sager, C. E. A theory of performance. In Schmitt, N., Bormann, W.C. et al. (Eds.), Personnel selection in organizations, 35-70, San Francisco, Jossey-Bass. 1993.
9. Feigenspan, J., Kästner, C., Liebig, J., Apel, S., & Hanenberg, S. Measuring programming experience. In Program Comprehension (ICPC), IEEE 20th International Conference on (pp. 73-82). IEEE. 2012.
10. Genero, M., Cruz-Lemus, J. A., Piattini, M.: Métodos de Investigación en Ingeniería del Software. RaMa 2014.
11. Sampieri, R. H., Collado, C. F., & Lucio, P. B. Metodología de la investigación. México: McGraw-Hill. 2010.
12. Kruger, J., & Dunning, D. Unskilled and unaware of it: how difficulties in recognizing one's own incompetence lead to inflated self-assessments. Journal of personality and social psychology, 77(6), 1121. 1999.
13. Wilking, D., Schilli, D., & Kowalewski, S. Measuring the human factor with the rasch model. In Balancing agility and formalism in software engineering (pp. 157-168). Springer Berlin Heidelberg. 2008.
14. Soriano, M. A. Algoritmos Voraces. Facultat d'Informàtica, U.P.C. Consultado el 3 de Enero de 2017, disponible en http://www.cs.upc.edu/~mabad/ADA/curso0708/GREEDY.pdf. 2007.
15. Deza, Elena; Deza, Michel Marie. Encyclopedia of Distances. Springer. p. 94. 2009.