La Salle Universidad

Revista de Ingeniería de Software

Vol 2 N° 1 Noviembre 2014

Ciencia y Tecnología

La perspectiva algorítmica es útil no sólo para la informática.
Dr. Marcos Kiwi

¿Y qué esperas tú?
Dr. (c) Jorge Poco

Lo que debe saber un ingeniero de Software.
Dr. Percy Pari Salas

Visualización Computacional como apoyo en tareas de Minería de Datos.
Mcs. Aurea R. Soriano Vargas

REVISTA DEL AREA DE COMPUTACION

Universidad La Salle de Arequipa

Simposio de Inteligencia Artificial (SPIA - 2014)

Como es tradición en los últimos años, se realizó en Arequipa el V Simposio Peruano de Inteligencia Artificial (SPIA) que en esta oportunidad fue organizado por el grupo de inteligencia artificial de la Sociedad Peruana de Computación (SPC) y se llevó a cabo el 06 y 07 de enero del 2014 en el auditorio Principal de la Universidad La Salle. El SPIA busca propiciar un intercambio académico entre la comunidad científica local, nacional e internacional, así como difundir los trabajos que están siendo realizados local, nacional e internacionalmente.

MARCOS

Lo que debe saber un ingeniero de software

Dr. Percy Pari Salas

Percy Pari Salas es doctor en ingeniería de software por la *Bond University* de Australia y tiene una maestría en ciencias de la computación por la Universidad de Sao Paulo en Brasil. Es especialista en Pruebas de Software, procesos de desarrollo de software y automatización de pruebas. En su artículo, el Dr. Percy Pari describe las características que debe tener un ingeniero de Software.

La perspectiva algorítmica es útil no sólo para la informática

Dr. Marcos Kiwi

Los algoritmos son importantes para cualquier sistema de software. Sin embargo, la algoritmia no es un tema que únicamente deban saber los profesionales en computación. En la siguiente sección, el Dr. Marcos Kiwi, quien obtuvo su Ph.D. en Matemáticas del MIT y es profesor titular en el Departamento de Ingeniería Civil Matemática e investigador asociado del Centro de Modelamiento Matemático de la Universidad de Chile, nos proporciona un artículo interesante sobre la perspectiva algorítmica.

Universidades de La Salle en el Mundo

Universidad La Salle México | Universidad La Salle Brasil UNILASALLE | Universidad de la Salle Bogotá | La Salle Universities | Escuelas Cristianas La Salle

ÍNDICE

La Salle Universidad

EDITOR
Dr. Cristian López Del Alamo
clopez@ulasalle.edu.pe

DISEÑADOR GRÁFICO
Jorge Luis Contreras Cano
http://ulasalle.edu.pe

2

La perspectiva algorítmica es útil no sólo para la informática

Marcos Kiwi es profesor titular en el Departamento de Ingeniería Civil Matemática e investigador asociado del Centro de Modelamiento Matemático, de la Universidad de Chile. Es Ingeniero Civil Matemático de la Universidad de Chile y Ph.D. en Matemáticas del MIT. Actualmente, es Editor Asociado de SIAM J. en *Discrete Mathematics y Theoretical Computer Science*. Sus principales áreas de investigación son la complejidad computacional, algoritmos aleatorios, y combinatoria

Dr. Marcos Kiwi

La perspectiva algorítmica es útil no sólo para la informática En ciencias de la computación, abundan ejemplos de como permean las nociones algorítmicas. Algunos de los cambios más significativos en los estándares de ruteo de la Internet pueden entenderse como debates sobre las deficiencias y beneficios relativos de uno u otro algoritmo para determinar caminos de costo mínimo. Los algoritmos juegan un papel cada vez más significativo en la determinación de qué información es la más relevante (especialmente en los motores de búsqueda para la Web). Sistemas de recomendación utilizan mecanismos algorítmicos para comparar nuestras preferencias contra otros miembros de nuestras redes sociales y en base a ello nos sugieren productos y actividades, nos "recuerdan" información, o inclusive nos dan cuenta de los "trending topics".

En un sentido amplio, los algoritmos son procedimientos para llevar a cabo una acción deseada a través de la realización de una secuencia ordenada de instrucciones. Estos procedimientos se refieren a un problema y proveen una forma de abordar su resolución.

Entendidos así, instrucciones paso a paso para la navegación pueden ser consideradas un algoritmo. De aquí la relevancia de la algorítmica no sólo para las Ciencias de la Computación. La siguiente frase de Bernard Chazelle, un profesor del Departamento de ciencias de la computación de la Universidad de Princeton, encapsula la referida idea de manera muy provocadora: "The Algorithm's coming-of-age as the new language of science promises to be the most disruptive scientific development since quantum mechanics."

A través de la historia, las matemáticas nos han proveído de un lenguaje extremadamente efectivo para describir el movimiento de cuerpos celestes, proyectiles y fluidos, evolución de reacciones químicas, la valoración de instrumentos financieros, etc. Los dominios en que el lenguaje matemático "reina" sin contrapeso son principalmente aquellos en que abundan las simetrías, la regularidad, la periodicidad, etc. Sin embargo, hay ámbitos de gran importancia que se resisten a ser entendidos a través de fórmulas y sistemas de ecuaciones (diferenciales o de cualquier otro tipo). Los biólogos entienden qué es y cómo se ve una célula, pero tienen mucho mayor dificultad para describir cómo es que se comportan.

Avances recientes en áreas de la biología tales como inmunología, genética moderna, embriología, y evolución, por nombrar algunas, reflejan la transformación desde una disciplina preponderantemente descriptiva a una centrada en la comprensión de procesos, ya sean procesos evolutivos de selección natural, replicación de ADN y proteínas, etc. El área de machine learning ha tenido un profundo impacto en áreas como la estadística, ampliando su enfoque desde uno basado en la elaboración de modelos analíticos de lo observado a una perspectiva basada en el diseño de procesos de clasificación, conceptualización y descubrimiento de la realidad observada.

En computación los procesos son ejecuciones de algoritmos. Desde su nacimiento, el concepto ha sido central para la informática. El pensamiento algorítmico es parte de la esencia misma de todo especialista en computación. Esto explica, en parte, la influencia e importante papel que dichos especialistas han tenido, el último par de décadas, en aquellas disciplinas que están transitando desde actividades descriptivas a otras enfocadas en entender procesos; especialmente procesos que comparten características propias de los algoritmos como recursión, paralelismo, auto-referencia, robustez ante fallas, auto-corrección, terminación, etc. La esperanza es que resulte útil el enfoque clásico de la especialidad consistente en preguntarse ¿cuán difícil es el problema?, ¿cuál es la forma más eficiente de resolverlo?, ¿cómo se descompone el problema en tareas más simples?, ¿cuáles son los recursos disponibles para su resolución?, etc., la esperanza es que sea fructífero usar abstracciones y métodos propios de la computación para elucidar por ejemplo la estructura de proteínas así como sus funciones, o comprender como se auto-organizan las bandadas de pájaros para volar minimizando el esfuerzo.

Por otro lado, es del todo válido afirmar que un algoritmo también puede ser visto como una función (de entradas en salidas) y que, por lo tanto, forma parte integral de las matemáticas, por lo que

no habría nada nuevo que la computación pudiera decir que no lo puedan hacer las matemáticas clásicas. Esto sería como sostener que excepto por la teoría de conjuntos, el resto de las áreas de las matemáticas no son una contribución importante dado que casi la totalidad de los objetos matemáticos de estudio son sólo conjuntos (más o menos complejos dependiendo de las propiedades que satisfacen dependiendo del objeto matemático que se trate).

Uno no puede dejar de dejarse seducir por la idea de ser uno de los rudimentarios conocedores del lenguaje llamado a develar el misterio de cuáles son los programas que determinan el comportamiento de las células, qué y cuándo realizar ciertas acciones, de qué proteínas y en qué secuencia deben interactuar para desencadenar procesos, ... si la computación nació con la construcción del primer computador multi-proposito, entonces recién estamos en la etapa de aprender a caminar para eventualmente liberar el real potencial del pensamiento y la perspectiva algorítmica.

Bibliografía

[1] Thomas A. Easton. Beyond the algorithmization of the sciences. pages 31–33, 2006.

[2] The Algorithm: Idiom of Modern Science. Chaotic dynamics for multi-value content addressable memory. http://www.cs.princeton.edu/~chazelle/pubs/algorithm.html. Accesado 28 de Marzo de 2014.

[3] Jeanette M. Wing. Computational thinking. pages 33–35, 2006.

■

¿Y qué espera tú?

Dr.(c) Jorge Poco es candidato a Doctor en el departamento de Ciencias de la Computación del *Polytechnic Institute of New York University* y trabaja en el área de Visualización de Datos

Dr.(c) Jorge Poco

Doctorado en Estados Unidos

En este artículo me gustaría comentar un poco sobre todo el proceso requerido para aplicar a un doctorado en cualquier parte del mundo y también explicar en detalle las peculiaridades de los doctorados en Estados Unidos. Creo que la forma más apropiada para poder contarles al respecto, es formulando algunas preguntas que muchos de ustedes deberían estar haciéndose si desean seguir el camino del doctorado.

¿Qué te anima a hacer un doctorado? Esta es la pregunta clave que un estudiante debe de saber responderse antes de comenzar este largo camino. Aquí en EUA, un doctorado promedio dura aproximadamente seis años, así que dos de los factores importantes son el tiempo que vas a invertir en esto y el estar alejado de tus familiares en un país donde la cultura es diferente. A mi parecer, uno necesita tener pasión por la investigación, porque a lo largo del camino te vas a encontrar muchos obstáculos y si tu convicción no es suficientemente fuerte, te van a llevar a desistir del doctorado.

¿Cuál es el proceso para poder postular a un doctorado en Estados Unidos? Normalmente todas las universidades en EUA comienzan sus procesos de admisión en Diciembre, para iniciar clases en Agosto del siguiente año. Esto quiere decir que hay que comenzar a trabajar en tu aplicación con al menos un año de anticipación. Los tres requisitos indispensables que toda universidad te pedirá son: puntajes de exámenes de inglés y conocimientos (TOEFL, GRE) y cartas de recomendación. Cada universidad define los puntajes mínimos para sus postulantes, así que en algunos casos tendrán que dar estos exámenes más de una vez antes de enviar sus papeles. Tengan en cuenta que tanto el GRE como el TOEFL tienen ciertas restricciones como: el número de veces que puedes darlo al año y el tiempo que deben esperar entre exámenes consecutivos. De la misma forma, las cartas de recomendación son de gran importancia porque será casi el único referente que los evaluadores tendrán específicamente sobre ti. Algo muy importante de enfatizar es que la falta de publicaciones no es un impedimento para postular, no es una exigencia, pero si las tienes es un bonus que te hará sobresalir entre los otros postulantes.

¿Qué te espera cuando comienzas tus estudios?

En esta pregunta existen muchas cosas que se pueden abordar, pero comencemos describiendo cómo son las evaluaciones a lo largo del doctorado. El estilo norteamericano es diferente al estilo europeo en los doctorados. Aquí todos los estudiantes de doctorado tienen que pasar sus dos primeros años tomando cursos generales sobre el área de Ciencia de la Computación. Esto no quiere decir que no harás investigación durante estos dos primeros años, al contrario, tendrás que dividir tu tiempo para llevar los cursos e investigar, pero la prioridad tendrán que ser tus cursos. Aproximadamente al finalizar el segundo año tienes que pasar un examen de cualificación. Este examen varía de universidad en universidad, en la mayoría de los casos este consiste de un examen escrito de conocimientos más un reporte de investigación, escrito totalmente por el alumno, el cual será defendido en una presentación delante de un jurado.

Durante los siguientes años te dedicas exclusivamente a tu investigación, para lo cual tienes que presentar una propuesta de tesis y finalmente defenderla. Aquí también es bueno mencionar sobre la ayuda económica. Esto es clave porque como estudiantes necesitamos una estabilidad económica para poder dedicarnos exclusivamente a los estudios. Con respecto a esto, en casi todas las universidades norteamericanas, los estudiantes de doctorado reciben apoyo financiero de dos formas: asistente de profesor o asistente de investigación. Normalmente cuando entras y no tienes asesor, entonces puedes trabajar como asistente de profesor ayudando a algunos de los profesores del departamento. Al finalizar el primer año todos los alumnos necesitan tener un asesor, una vez que este esté definido, entonces pasarás a ser un asistente de investigación, trabajando en los proyectos de investigación de tu asesor.

Con relación a los asesores, profesores que te guían hasta el término de tu tesis, su forma de trabajo varía de persona en persona. Hay algunos a quienes les gusta participar directamente en los detalles de cada proyecto, así como hay algunos que solo esperan que llegues a conversar con él cuando tienes algún resultado. Esto es importante y puede ser la clave para concluir o desistir del doctorado. Entonces, antes de escoger un asesor, pregúntense que clase de investigadores son ustedes. Resumiendo lo descrito anteriormente, uno tiene que tener vocación de investigador antes de comenzar este camino. Si ustedes deciden comenzarlo, entonces hay que prepararse con tiempo. Asimismo, a lo largo de sus estudios se encontraran muchas dificultades, la clave esta en no desistir y no cerrarse en una sola forma de trabajo, hay que probar diferentes alternati-

vas hasta encontrar la mejor para nosotros. Finalmente, me gustaría recomendarles la lectura del libro publicado por Philip Guo de la Universidad de Stanford "THE PH.D. GRIND, A Ph.D. Student Memoir" `http://www.pgbovine.net/PhD-memoir.htm` que relata su experiencia durante sus años de estudio de doctorado. ■

2

Lo que debe saber un ingeniero de software

Percy Pari es Doctor en Ingeniería de software por la *Bond University* de Australia y tiene una maestría en ciencias de la computación por la Universidad de Sao Paulo en Brasil. Es especialista en Pruebas de software, procesos de desarrollo de software y automatización de pruebas

Dr. Percy Pari Salas

Para poder definir qué es lo que un Ingeniero de software debe saber, primero debemos ser capaces de definir que es lo que es un Ingeniero de software o, al menos, definir cuál es su perfil. Yo entiendo, y comparto, la posición de que la ingeniería de software no es una ingeniería en el sentido tradicional. Sin embargo, mantengo que en un sentido abstracto la intención es prestarse conceptos que son bien entendidos en la ingeniería tradicional para ser adaptados y aplicados al software. Así, por ejemplo, los ingenieros, en general, son vistos como los constructores (o responsables por la construcción) de estructuras o proyectos complejos

Los civiles construyen puentes y edificios, los ingenieros de materiales crean (construyen) estructuras químicas que le dan diferentes propiedades a los materiales, los ingenieros mecánicos construyen motores o máquinas complejas, etc.

Siguiendo esta analogía, mi opinión personal es que, de lo que se trata de describir con el término Ingeniero de software, es a un desarrollador (constructor) de software. La pregunta cae por su propio peso, ¿un Ingeniero de software es un Programador? por que no le llamamos Programador y nos evitamos la confusión que crea el uso del termino Ingeniero?

La respuesta exige un poco de análisis en cuán adecuada es nuestra analogía. Primero, como dije antes, la idea de llamarle ingeniería (al desarrollo/construcción de software) tiene como objetivo aplicar (y adaptar) conceptos de la ingeniería tradicional al desarrollo de software. Como la historia dice, el término ingeniería de software nace del deseo de incrementar la calidad del software y de su proceso de desarrollo.

La calidad (en la ingeniería tradicional) se asegura midiendo y controlando los procesos de construcción y sus productos. Estas medidas se dan sobre ciertas propiedades físicas (o naturales en general, químicas, por ejemplo) que presentan los artefactos que son construidos. Estas propiedades son bien entendidas y estudiadas por las ingenierías tradicionales. Sin embargo, en relación al software, estas propiedades no son simples (a veces son abstractas) ni están maduras y muchas veces no son bien entendidas.

De ahí que un Ingeniero de software debe ser un Programador con una formación adicional en cuanto se refiere a Métricas de software, Aseguramiento de la Calidad, Calidad de Procesos, etc.

Es cierto también que hay **programadores** y programadores. Es decir, hay programadores que no requerirían de la ingeniería de software para garantizar la calidad de su trabajo, sino que, básicamente, les nace producir software de buena calidad. En realidad, son Ingenieros natos que trabajan como programadores y que probablemente no han sido formalmente entrenados en una Facultad de ingenierías. Pero, como estos programadores (estrella) son escasos en el mercado, el mercado necesita habilitar programadores (promedio) que realizan en general un buen trabajo pero que pueden tener algunos altibajos.

Es ahí, donde el rol de la ingeniería de software es importante, habilitar a programadores promedio para producir software de buena calidad de manera consistente. De ahí que el Ingeniero de software necesita saber de Procesos Cognitivos y necesita aprender los lineamientos básicos de cómo trabajar en equipo (Dinámica de Grupos probablemente) y de cómo transmitir el conocimiento eficientemente.

La demanda del mercado también juega un papel importante en la definición de lo que un Ingeniero de software debería saber. La analogía del Ingeniero de software como constructor de productos se rompe cuando vemos que gran parte de la demanda en el campo del software es por servicios más que por productos. Google, por ejemplo, basa su subsistencia económica en la oferta de servicios (de búsqueda, de correo electrónico, de anuncios publicitarios) más que en la venta de sus productos (si es que vende alguno). Es así que un Ingeniero de software también necesita saber de procesos organizacionales que le permitan soportar el uso de un producto de software para brindar servicios. Ciertamente, lidiar con servicios de software es una parte de la ingeniería de software que está más relacionada al área de los Sistemas de Información.

No olvidemos, también, el concepto de calidad; el cual juega un papel importante, especialmente en el software. La

definición de calidad más aceptada actualmente está centrada en el concepto de *fitness for use*; es decir, que la calidad es medida en términos de cuán eficientemente se atienden las necesidades del usuario. El problema de la calidad en el software es que por su naturaleza abstracta los usuarios tienden a percibirlo como un artefacto sumamente adaptable. Pongamos un ejemplo, difícilmente una persona pretendería usar un martillo como si fuese un destornillador, o pretendería llevar su martillo a un herrero para que lo convierta en un destornillador. Sin embargo, cuando hablamos de software, los usuarios son mucho más proclives a cambiar sus requerimientos de formas bastante radicales. La ingeniería de software intenta atacar este problema definiendo métricas que le permitan a los desarrolladores estimar y demostrar el costo de estos requerimientos de una manera convincente y entendible para los usuarios.

Un Ingeniero de software, además de conocer las métricas existentes, debería ser capaz de crear o reconocer otras métricas que sean adecuadas a situaciones particulares y, además, debería ser capaz de manejar las expectativas de los usuarios, de manera que, cuando el desarrollo de un producto de software llegue al punto donde este es entregado al usuario, la calidad percibida del producto no sea decepcionante.

Por lo expuesto hasta ahora, me atrevo a definir el perfil del Ingeniero de software como:

Un Programador que es capaz de producir software de calidad consistentemente, es capaz de habilitar a otros programadores a continuar con el desarrollo de software iniciado por él, o de dar soporte a productos de software previamente desarrollados por él, y es capaz de continuar o mantener el desarrollo de otros programadores (ingenieros de software).

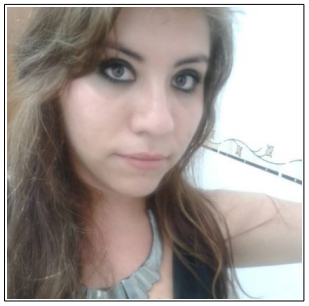
De esta definición general se desprende que el Ingeniero de software es capaz de reconocer software de calidad (y aquél que no lo es) y de demostrar esta apreciación de manera objetiva, con métricas consistentes, sobre las propiedades del software. Finalmente, el Ingeniero de software no debe ser sólo técnicamente capaz, sino que, también, debe ser capaz de comunicar efectivamente a los usuarios del software y a los clientes (aquellos que pagan por este desarrollo) los costos, riesgos, dificultades técnicas y de procesos en el desarrollo de dicho software.

Un Ingeniero de software debe, siempre, tener en cuenta que debido a que las propiedades del software son, en general, abstractas y producto de una pobre comunicación con el usuario, en el desarrollo (construcción) de software, se corre el riesgo de que la endeble calidad del producto no sea percibida a tiempo, con el consecuente costo que implica el desarrollo o el uso de un software de baja calidad. ■

2

Visualización Computacional como apoyo en tareas de Mineración de Datos

Aurea R. Soriano Vargas es Magíster en Ciencias de la Computación de la Universidade de São Paulo-ICMC y estudiante de Doctorado de la misma universidad.



Aurea R. Soriano Vargas.

El avance rápido de la tecnología, tanto en *software* como en *hardware*, permite que los usuarios tengan mayores opciones de expandir las capacidades de producción, comunicación e investigación. Mientras tanto, inmensas cantidades de datos de alta dimensionalidad están siendo generados y almacenados, cayendo en la llamada "maldición de la dimensionalidad"[7]. Explorar estos conjuntos de datos involucra la dificultad de identificar los atributos que son realmente relevantes para su análisis, ya que trabajar con todos, sean relevantes o no, tienen un impacto significativo en la complejidad del análisis y en el costo computacional de los métodos[1]. Esta dificultad representa un desafío sin precedentes en el área de Mineración de Datos[5]. El enfoque clásico para minimizar la influencia de la "maldición de la dimensionalidad" es usar técnicas de reducción, las cuales se dividen en extracción y selección de atributos. La Figura 1 muestra los diferentes enfoques de reducción de dimensionalidad.

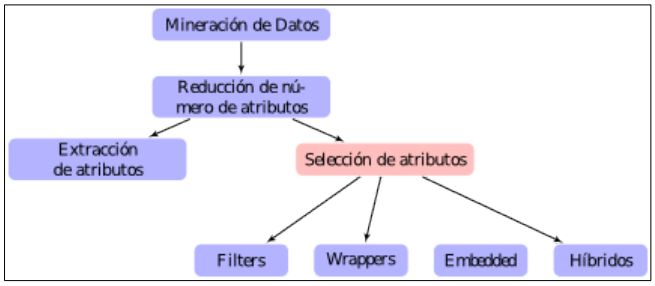


Figura 1: Enfoques de reducción de atributos

Las técnicas de extracción de atributos, crean un nuevo conjunto de menor dimensionalidad, mediante transformaciones lineales o no lineales, sin preservar la naturaleza de los atributos originales. Por el contrario, la selección de atributos preserva aquella naturaleza, lo que permite explicar de forma intuitiva el resultado. Por ello, existe un especial interés en el estudio de técnicas de selección de atributos. La selección de atributos puede ser definida como el proceso de escoger un subconjunto de atributos que represente la información importante del conjunto original, según algún criterio [8]. La Figura 2 exhibe el proceso de selección al detalle. Ese subconjunto puede ser de igual tamaño o de menor (denominado dimensión intrínseca) al conjunto de atributos original (denominado dimensión de inmersión). Cabe resaltar, que la dimensión de inmersión y la dimensión intrínseca son iguales para los conjuntos que cumplen propiedades de uniformidad e independencia, o sea, conjuntos de atributos independientes entre sí y cuyos valores son distribuidos de manera uniforme o aleatoria. Caso contrario, si existe alguna correlación entre los atributos, la dimensión intrínseca será menor a la dimensión de inmersión[11]. Es importante mencionar que aquel subconjunto es el resultado de descartar aquellos atributos irrelevantes o redundantes. Se consideran atributos redundantes, aquellos que contienen valores correlacionados, los cuales se han descubierto que afectan la precisión de los clasificadores y por lo tanto deberían ser eliminados[4].

Figura 2: Proceso de selección de atributos según Liu (2005)[9]

La mayoría de los métodos tradicionales de selección de características presentan una naturaleza de "caja negra", esto hace imposible que el usuario pueda seguir el proceso completo, convirtiendo estos métodos en poco comprensibles, además de no considerar el conocimiento de usuarios expertos. En ese contexto, nuevos métodos basados en Visualización de Información han sido propuestos para tratar estas limitaciones. Estos métodos permiten apoyar a las personas en los procesos que necesitan ser comprendidos y que requieren extraer informaciones por medio de interacciones con el usuario a través de representaciones gráficas de los datos. Las interfaces visuales pueden contribuir significativamente en una ejecución exitosa de tareas de extracción de conocimiento a partir de datos, además de otras tareas de adquisición de conocimiento de naturaleza menos sistemáticas y más exploratoria, aprovechando la capacidad visual del ser humano de detectar y reconocer patrones de manera más fácil y rápida[6]e de la participación activa del usuario en el proceso. Esta sinergia entre Mineración de Datos en Aprendizaje de Máquina y Visualización de Información[3] ha sido explorada en Mineración Visual de Datos [2,10] que busca integrar la capacidad humana de reconocimiento de patrones visuales a procesos automáticos o semi-automáticos de mineración tradicionales. Especialmente, la visualización procura ofrecer a los analistas un "modelo mental" adecuado del universo de los datos que se están manipulando, capaz de favorecer la percepción de las características globales, y la identificación y exploración interactiva de sus particularidades.

Bibliografía

[1] Kevin Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. When is nearest neighbor meaningful. In Database Theory ICDT, pages 217–235. Springer, 1999.

[2] Katy Börner, Chaomei Chen, and Kevin W. Boyack. Visualizing knowledge domains. Annual Review of Information Science & Technology, 37:179–255, 2003.

[3] Stuart K. Card, Jock D. Mackinlay, and Ben Shneiderman. Readings in information visualization: using vision to think. Morgan Kaufmann Publishers Inc., 1999.

[4] Mark A. Hall. Correlation-based feature selection for discrete and numeric class machine learning. In Proceedings of the Seventeenth International Conference on Machine Learning, pages 359–366, 2000.

[5] Jiawei Han, Micheline Kamber, and Jian Pei. Data mining: concepts and techniques. Morgan kaufmann, 2006.

[6] Christopher G. Healey, Kellogg S. Booth, and James T. Enns. Visualizing real-time multivariate data using preattentive processing. ACM Trans. Model. Comput. Simul., 5(3):190–221, 1995.

[7] F. Korn, B.-U. Pagel, and C. Faloutsos. On the ldquo;dimensionality curse rdquo; and the ldquo;self-similarity blessing rdquo;. Knowledge and Data Engineering, IEEE Transactions on, 13(1):96–111, 2001.

[8] Huan Liu and Hiroshi Motoda. Feature selection for knowledge discovery and data mining. Springer, 1998.

[9] Huan Liu and Lei Yu. Toward integrating feature selection algorithms for classification and clustering. Knowledge and Data Engineering, IEEE Transactions on, 17(4):491–502, 2005.

[10] Maria Cristina Ferreira Oliveira and Haim Levkowitz. From visual data exploration to visual data mining: A survey. IEEE Transactions on Visualization and Computer Graphics, 9(3):378–394, 2003.

[11] E. P. M. Sousa. Identificação de correlações usando a Teoria dos Fractais. PhD thesis, Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2006.

■

4

Entre el mundo académico y el empresarial

Christian Danniel Paz Trillo, es Magíster en Ciencia de la Computación por el Instituto de Matemática y Estadística de la Universidad de São Paulo y con cerca de diez años de experiencia en Desarrollo de Software, actualmente es Especialista en Servicios de Autenticación en RENIEC

Christian Danniel Paz Trillo

Desde que salí de la universidad he tenido la oportunidad de navegar entre el mundo académico y el empresarial. Puedo decir que conozco los beneficios y los sacrificios que ambos mundos nos ofrecen así como los desafíos que ambos nos imponen. En este texto, intentaré transmitirles mi percepción de como se presenta el mundo empresarial para alguien después de la formación que recibe en la universidad.

En primer lugar, un título, universitario o no, te abre puertas, es tu carta de presentación. Creo que sabiendo explicar tu formación y tu perfil, el hecho de haber estudiado Ingeniería de Sistemas, Sistemas de Información o Ciencia de la Computación, no hace que te discriminen de una determinada oferta laboral, por lo menos yo no discriminé cuando participé en procesos para seleccionar personas.

Difícilmente al salir de la universidad, uno pueda exigir hacer sólo lo que uno quiere. Después de terminar mi maestría, trabajé en una empresa que iniciaba sus actividades en el país en que me encontraba, y el único informático (como me llamaban) en la sede era yo. A pesar de que no era mi actividad favorita, tuve que configurar servidores, instalar bases de datos, configurar la impresora cuando había problemas. Aprendí mucho, y hasta hoy, es un conocimiento que me permite ganarme la confianza de la gente con que trabajo.

Sin embargo, poco a poco se hizo evidente que tener a alguien con mi perfil en esas actividades era ineficiente, porque no era mi especialidad y otros lo harían mejor, y porque mi tiempo podría ser mejor aprovechado en otras actividades. Esto me enseñó que para tener éxito uno tiene que ser comprensivo, adaptarse a la realidad de la organización. A modo de incentivo, al salir de la empresa ya había un equipo que se encargaba de cuidar de la infraestructura y otro dedicado al desarrollo de software.

A continuación listaré algunos aspectos básicos que le recomiendo tener en cuenta a los que deseen trabajar en desarrollo de software en el mundo empresarial:

Un desarrollador tiene que ser capaz de justificar sus decisiones técnicas, estar seguro de las mismas y si es posible o necesario, hacer que sean respaldadas por el responsable técnico del proyecto de software. La justificación puede ser por un respaldo de experiencia en la organización (estándares), por una experiencia propia que pueda ser demostrada, o por alguna referencia externa de confianza: artículos, proyectos o comunidades. Sin esto, una decisión técnica corre serios riesgos de ser cuestionada a posteriori y, como hemos oído muchas veces, los cambios en etapas avanzadas del desarrollo son muy costosas.

El profesional de desarrollo debe ser capaz de identificar los riesgos en un desarrollo: riesgos organizacionales y riesgos técnicos.

- Los riesgos organizacionales se refieren a cosas que dependen de otras personas o áreas de la organización y que pueden afectar algún aspecto del desarrollo. Algunos ejemplos son: disponibilidad de algún dispositivo, instalación y configuración de un servidor o desarrollo de un componente por algún otro equipo.
- Los riesgos técnicos son actividades que se tendrán que desarrollar y que, por algún aspecto técnico, no hay certeza total de que vayan a dar los resultados esperados en el plazo esperado. Algunos ejemplos son: uso de bibliotecas nuevas, integración entre plataformas, o cosas, que la documentación dice que se pueden realizar pero, con las cuáles no se tiene experiencia.

Para poder garantizar que se cumplan los plazos, uno tiene que minimizar estos riesgos. Crear pruebas de concepto para eliminar los riesgos técnicos, es un primer paso fundamental en cualquier desarrollo complejo.

Recomiendo fuertemente también preguntar y cuestionar. Preguntar es fundamental para aprender, para no reinventar la rueda, para reutilizar lo que ya existe en la organización. Pero no sólo eso, preguntar te permite también dar mayor relevancia al trabajo del otro, lo incentivas y le muestras que lo que él hace le es útil a otras personas en la organización. Cuestionar también es importante. Cuando se recibe una especificación que involucra una decisión técnica ya tomada, es necesario pensar en las alternativas y verificar que la decisión fue realmente la mejor. Si consideramos que hay soluciones mejores, debemos cuestionar, con el fin de entender la decisión. Si la solución que consideramos mejor fue analizada, entender por qué no fue escogida. Si la solución no fue analizada, cuestionar si la decisión podría ser cambiada justificando la propuesta. En el peor de los casos, recibiremos una negativa, normalmente justificada, pero no nos quedaremos con la duda.

No reinventar la rueda también es fundamental en un contexto empresarial. En la universidad tenemos que hacer todo desde cero porque nuestro objetivo es aprender. El objetivo de las organizacio-

nes es ser productivo. Si ya existe algo, por ejemplo una biblioteca, en otro proyecto que hace lo que necesitamos hacer, es necesario evaluar si ese algo está bien hecho, en todos los aspectos: código, eficiencia, documentación y pruebas. Si está bien hecho, reutilizarlo. Si le falta algo, mejorarlo y hacerlo disponible para el proyecto original, así nuestro esfuerzo tiene un impacto mayor. Si tenemos que desarrollar algo desde cero, pero consideramos que lo que estamos haciendo puede ser reutilizado en el futuro, debemos hacerlo de forma que sea fácilmente reutilizable y divulgarlo.

Si existen bibliotecas o proyectos de código abierto, con las licencias adecuadas, que hagan lo que necesitamos, también recomiendo reutilizar. Un ejemplo clásico sería el de reimplementar un algoritmo de ordenación que ya está disponible en cualquier biblioteca. Sin duda existen casos en que es necesario hacerlo, cuando se requiere alguna implementación muy específica que tiene que optimizar algún aspecto de los elementos que están siendo ordenados, pero en la mayoría de los casos no es necesario. Esto no significa que no nos importe la eficiencia, debemos verificar que la biblioteca hace una implementación adecuada a nuestras necesidades.

Un profesional tiene que valorizar su trabajo, mostrar que la actividad de la programación es compleja, no es "sólo programar". He visto muchos desarrolladores que se inician y que reciben tareas con plazos irreales y las aceptan por temor. Se les presenta un desafío y sin medir su complejidad aceptan los plazos impuestos. Siempre hay que recordar las tres variables que conforman el triángulo que afectan la calidad del desarrollo de software: tiempo, recursos y alcance. Dada una tarea con un determinado alcance, si no tenemos el tiempo suficiente, no lo haremos con la calidad necesaria. Si es que hay un plazo muy estricto, debemos negociar el alcance, proponer una versión reducida que sea suficiente para cumplir lo que se espera en el plazo establecido.

Finalmente, es importante tener claro el perfil que uno busca. Personalmente, me gusta ser un desarrollador de soluciones de software: diseño de arquitectura, implementación, pruebas y despliegue. Cuando se presenta un desafío técnico: plantear una solución integral, coherente, viable y dentro de los estándares de la organización. Otros profesionales prefieren las fases previas como el levantamiento y análisis de requisitos, fases tan necesarias como las posteriores. Otros prefieren la especialización en determinados lenguajes, plataformas o frameworks. En términos económicos, algunos perfiles son más valorizados que otros, así que uno tiene que contrapesar lo que a uno le gusta hacer, lo que uno sabe hacer y lo que uno necesita hacer para tener una buena remuneración. Con esas tres variables en mente, podemos definir nuestro perfil. ■

2

Mujeres de ciencias de la computación trabajando en la empresa: Experiencia

Claudia Talavera Garnica trabaja hace 2 años en *Tata Consultancy Services*, en el puesto de *Developer*, y actualmente está en el proyecto Rímac Seguros realizando tareas de extracción y minería de datos, donde está a cargo del Proyecto de Migración de Pólizas Desgravamen. Sus áreas de interés son Inteligencia de Negocios y Base de Datos

Claudia Talavera Garnica.

No es necesario hacer un estudio de mercado para darnos cuenta que el género femenino es una minoría cuando hablamos de Ciencias de la Computación. Aunque muchas señales indiquen que es una carrera dominada por varones, creo firmemente que una mujer puede hacerse camino y tener mucho éxito en este campo. Recuerdo la primera vez que ingresé al salón de clases, me encontré con un intenso aroma masculino y es que ¡se respiraba testosterona! Solo éramos un pequeño grupo de mujeres, pero esto no nos amilanó o hizo que nos rindiéramos, sino que fue un impulso para demostrar que Ciencias de la Computación no es una cuestión de género sino de habilidades puestas en práctica.

A través de los años de estudio, fui alimentando mi creatividad y capacidad de análisis. Esta profesión te enfrenta a un problema nuevo cada día, un obstáculo diferente, un nuevo lenguaje; la rutina es casi nula, pues cada día supone una solución tecnológica diferente. Y si es que existe un día tranquilo, ya estamos pensando en mejoras en nuestra implementación. El mundo de Ciencias de la Computación es amplio. Se puede ejercer la carrera en diferentes ramas. Existe un sinfín de oportunidades de trabajo y estudio. No hay obstáculos que impidan desarrollar tu potencial.

Mi experiencia profesional se inició hace cuatro años en mi alma máter. Me dediqué a tareas de apoyo en el análisis y desarrollo de requerimientos. Posteriormente, tuve la oportunidad de ser parte de una fábrica de software, donde el diseño e implementación se llevan a cabo en un terreno más ordenado. Aprendí mucho sobre metodologías, estimación, disponibilidad de aplicaciones, sin mencionar que revisé aplicaciones que iban desde COBOL, pasando por Power Builder hasta C y Javascript.

Queriendo afrontar nuevos retos en mi carrera tome la decisión de mudarme a otra ciudad. Actualmente trabajo en Tata Consultancy Services, una empresa transnacional fundada en India que brinda servicios de tecnología a diferentes empresas y rubros. El proyecto en el que me desenvuelvo hoy en día es en una empresa aseguradora. Estoy a cargo de la migración de pólizas de AS400 a Oracle 11g.

Las mujeres tenemos cualidades que nos hacen óptimas para el trabajo en cualquier rama de Ciencias de la Computación. Somos detallistas, analíticas, observadoras por naturaleza. Esta es una carrera que requiere de capacidad de comunicación, empatía, innovación y liderazgo. Quien mejor que una mujer para poder interactuar con equipos de trabajo y usuarios; entender las necesidades del negocio y liderar equipos.

Pues Ciencias de la Computación no es estar todo el día sentado frente a un computador, sino que es una carrera que nos obliga a estar en contacto con otras personas, entender sus necesidades y procurar una mejor calidad de vida a través de los sistemas que podamos diseñar e implementar. Como mujeres, hacemos de la computación una carrera más humana. ■

2

Java 8: un nuevo presente para la plataforma

Paul Mendoza del Carpio es Magíster en Ciencias con mención en Ingeniería de Software por la Universidad Nacional de San Agustín. Docente Investigador en la Universidad La Salle. Profesional IBM Certificado enfocado en arquitecturas sobre la plataforma Java EE

Paul Mendoza del Carpio.

Oracle inició un largo camino en el año 2010, cuando el JCP (Java Community Process) votó a favor de la especificación Java 8. Y en el mes de marzo del presente año, finalmente se liberó la nueva versión de Java, la cual es la actualización más significativa realizada sobre el lenguaje.

Java fue diseñado en los 90's como un lenguaje de programación orientado a objetos, por esos años la programación orientada a objetos era el principal paradigma de desarrollo de software. Recientemente el interés en la programación funcional, más allá de su uso en el campo académico, ha crecido gracias a su buen manejo de concurrencia y eventos[3]. Lo anterior no implica que la programación orientada a objetos sea deficiente, más bien resulta una buena alternativa combinar ambos paradigmas para contar con lo mejor de cada uno. Para muchos, la principal mejora realizada sobre Java 8 es el agregado de nuevos elementos obtenidos de la programación funcional.

Entre los elementos y mecanismos nuevos presentes en Java 8 se tienen: expresiones Lambda, interfaces funcionales, métodos por defecto, anotaciones de tipos, referencias a métodos, uso de streams, API mejorada para Date y Time (incluye internacionalización), actualización de la API Collection (uso de expresiones Lambda y streams), eliminación del espacio PermGen [5,6,7].

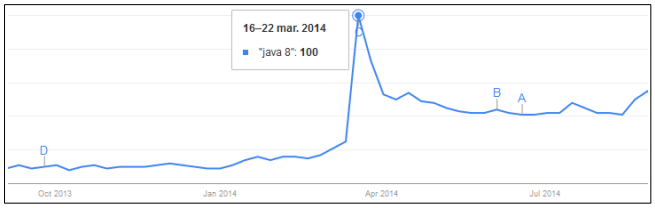
Entre los principales beneficios que trae consigo el uso de Java 8 se tienen:

- Mayor productividad: el uso de *Lambda Expressions* cuenta con mecanismos que permiten reducir el código repetitivo. Luego, el código puede ser más compacto y sencillo, lo cual puede incrementar la productividad de los desarrolladores en Java 8.
- Plataforma embebida: los desarrolladores en Java 8 cuentan con *profiles* en Java 8 para la implementación de aplicaciones en dispositivos con restricciones respecto a recursos.
- Integración con *JavaScript*: Java SE 8 incluye un motor de *JavaScript* que se ejecuta sobre la máquina virtual y que permite a las aplicaciones Java contener componentes escritos en *JavaScrip*t. Ello tiene en consideración la interoperabilidad y buen performance entre código Java y *JavaScript*.

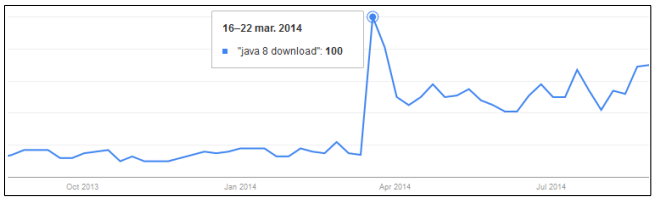
Cabe señalar la participación de importantes organizaciones en el desarrollo y evolución de la plataforma Java, donde han contribuido representantes de empresas como IBM, Intel, Red Hat, SAP. Java 8 presenta nuevas capacidades que extienden el popular lenguaje Java hacia una era de despliegue de aplicaciones en la nube. Al respecto, Mike Peach, gerente general de middleware en Red Hat menciona[4]: ***"The big thing in Java 8 is of course project Lamda, which enables developers to more effectively use concurrency and the callback style of programming popular in cloud development".*** Acerca del beneficioso uso de expresiones Lambda, James Donelan, vice-presidente de ingeniería en MuleSoft, opina lo siguiente[4]: ***"This is really significant, as functional programming has made a big comeback recently with more developers moving to JavaScript, Scala and Clojure due to their expressiveness and support for functional programming".***

Asimismo, Donelan indica que la capacidad de poder ejecutar código JavaScript en el estable JVM, puede formar una nueva clase de desarrolladores en la plataforma Java[4]. Las nuevas características de Java pueden impactar en los mecanismos que los desarrolladores emplearán para el acceso a datos en aplicaciones empresariales. Nuevas herramientas y bibliotecas de Java podrán ser desarrolladas para simplificar la ejecución de sentencias sobre bases de datos. Un ejemplo de ello es el proyecto Jinq, el cual permite emplear streams y un API fluido para consultas sobre bases de datos[1]. Entre las ventajas de emplear un API como la mencionada se tienen: evitar código repetitivo, verificación de errores.

Actualmente, aquellos desarrolladores que deseen empezar a utilizar el JDK 8 en aplicaciones Java Web, pueden empezar a utilizarlo con servidores web como Tomcat, Jetty y WildFly, los cuales presentan compatibilidad. Cabe señalar que el popular Spring Framework en su versión 4, ha sido diseñado desde su creación considerando Java 8, y ya cuenta con una liberación compatible también[2]. Finalmente, a fin de apreciar el actual interés en Java 8 en el mundo, se proporcionan imágenes con unos indicadores de frecuencia de búsqueda para Java 8, obtenidos desde Google Trends para el intervalo de meses de agosto 2013 a agosto 2014. Nótese que los picos más altos de búsqueda se alcanzaron en el mes de marzo del 2014, mes de la liberación de Java 8, luego las frecuencias de búsqueda se han mantenido más elevadas que antes de su liberación.



Frecuencia de búsqueda de Java 8



Frecuencia de búsqueda de Java 8 download

Bibliografía

[1] Lukas Eder. Java 8 friday: Java 8 will revolutionize database access. http://www.javacodegeeks.com/2014/03/java-8-friday-java-8-will-revolutionize-database-access.html.

[2] Juergen Hoeller. Java 8 in enterprise projects. http://spring.io/blog/2014/03/21/java-8-in-enterprise-projects.

[3] Cay S. Horstmann. Lambda expressions in java 8. http://www.drdobbs.com/jvm/lambda-expressions-in-java-8/240166764.

[4] Sean Kerner. Java 8 officially released, modularity still a concern. http://www.datamation.com/open-source/java-8-officially-released-modularity-still-a-html.

[5] Ted Neward. Java 8: Lambdas, part 1. http://www.oracle.com/technetwork/articles/java/architect-lambdas-part1-2080972.html.

[6] Venkat Subramaniam. Functional Programming in Java: Harnessing the Power of Java 8 Lambda Expressions. The Pragmatic Programmers, 2014.

[7] Richard Warburton. Java 8 Lambdas. O'Reilly Media, 2014.

■

Espacio dedicado a estudiantes

Noticias Cortas

Información después de la muerte

Escrito por César Calle Espino.

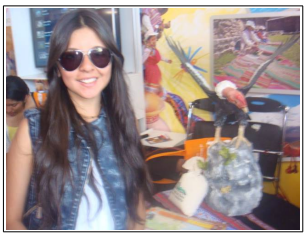
Imagínate que de un día para el otro dejas atrás este mundo terrenal y falleces. Tarde o temprano tendrá que pasar. ¿Y qué pasa con todas tus pertenencias? En los testamentos se especifican los deseos del fallecido después de morir; y como su última voluntad, también incluye a los herederos de sus pertenencias.

Ahora bien, el mundo avanza a un alto ritmo de innovación y digitalización. Podemos darnos cuenta que gran parte de nuestra vida está ligada a las redes sociales y servicios de almacenamiento; pero una vez que morimos, ¿Qué es lo que sucede con toda esta información?, también es parte de tu legado. Los diversos servicios están implementando políticas para luego del fallecimiento, en donde el usuario decide qué hacer con su información. Por ejemplo: Youtube, Hotmail, MySpace, Facebook y Linkedln conservan la información del usuario a menos que un allegado presente un documento que certifique el fallecimiento del mismo. En cambio Gmail desactiva una cuenta luego de 9 meses de inactividad.

Además actualmente existen servicios para que decidas que hacer con tu información luego de fallecido. Alguno servicios que se presentan son: Legacy Locker, Dead Man's Switch, Secure Safe, EZ-Safe, AssetLock. Todos son gratuitos excepto el último, que tiene un pago de 10 a 240 dólares. Legacy Locker es un repositorio donde se incluyen fotos e información importante para el usuario al que puede acceder tus familiares. Dead Man's Switch envía correos póstumos a tus contactos (se activa luego de la inactividad). Secure Safe tiene poco espacio de almacenamiento pero hasta 50 contraseñas.

E-Z-Safe permite dejar a un heredero de tu información. AssetLock almacena información, contraseñas e instrucciones para un usuario designado que se habilita después de la muerte.

Mujeres Navegando en una internet de hombres



Escrito por Katerine Arenas Torres.

La mayoría de personas queramos o no tenemos ciertos tabús en relación a varias cosas y hay uno en particular que debería llamar la atención de todos: número escaso de féminas en territorio masculino. Pese a que parezca poco importante, existe una infinidad de jóvenes mujeres que cada año tienen la interrogante de no saber que estudiar, aunque la mayoría no tomaría la opción de carreras similares a ingenierías de programación, ya que aun sobreviven algunos prejuicios referidos a que no es para ellas, igual que: "las mujeres tienen poco protagonismo en estos campos", "¿una chica en Informática?", "¿una mujer que estudia Software?", entre otros que se reconoce como propiedad de varones que desgraciadamente es lo que se cree. Aparentemente esto es cierto, pero a lo largo de la historia ha habido mujeres que han incursionado en estas materias que se creía exclusivamente para hombres, claro ejemplo es Ada Byron (hija del poeta Lord Byron) quien demostró su iniciativa y esfuerzo al ser la primera programadora. Las carreras que conllevan solucionar problemas con algoritmos representan un gran esfuerzo mental, de igual manera cualquiera puede programar, sin embargo ¿es el esfuerzo o atrevimiento que se requiere lo que asusta a las mujeres y les impide estudiar una carrera similar?, la solución más convincente resulta ser la audacia.