FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS UNIVERSIDAD DE CHILE

REVISTA

BITS de Ciencia

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

UNIVERSIDAD DE CHILE

Nº 3 / Segundo Semestre 2009

Educación en Informática

REVISTA
BITS de Ciencia
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN
UNIVERSIDAD DE CHILE

Dirección

Departamento de Ciencias
de la Computación
Avda. Blanco Encalada 2120, 3º piso
Santiago, Chile.
837-0459 Santiago
www.dcc.uchile.cl
Teléfono: 56-2-9780652
Fax: 56-2-6895531
dcc@dcc.uchile.cl

Revista Bits de Ciencia N° 3
ISSN 0718-8005 (versión impresa)

www.dcc.uchile.cl/revista
ISSN 0718-8013 (versión en línea)

CONTENIDOS

EDITORIAL

La educación en Ciencia de la Computación e Informática ha sido históricamente la mayor preocupación del Departamento (DCC). De hecho, uno de nuestros principales nortes -como está explícito en la Misión- es entregar educación superior integral y de excelencia en esta área.

Sin embargo, como es evidente, la disciplina ha cambiado dramáticamente en los últimos 15 ó 20 años. El advenimiento de la Internet y de una multitud de nuevas tecnologías y aplicaciones que hacen que el mundo se vuelva cada vez más conectado, y la gente más familiarizada con los computadores y la computación, ha obligado a replantear el concepto de Informática desde sus fundamentos. La Ciencia de la Computación hoy en día, y mucho más que antes, es un ente fantásticamente dinámico que se nutre para su estudio de un sinnúmero de problemas del mundo real; aquellos que aparecen día tras día en la práctica de la Computación. Todo esto implica que el experto en Informática debe ser, con especial énfasis, un profesional capaz de adecuarse a los cambios, de proponer soluciones y desafíos. Pero para lograr formar a este nuevo tipo de profesional es indispensable adecuar la educación de esta disciplina a los actuales requerimientos.

El tema central del presente número de la revista Bits de Ciencia intenta dar una breve -aunque significativa - visión sobre las nuevas maneras de enseñar Informática. Para ello hemos invitado a varios expertos nacionales, más dos invitados internacionales, a contarnos sus experiencias. Es el caso de Ernesto Cuadros-Vargas, de la Sociedad Peruana de la Computación, quien nos habla del trabajo realizado con su grupo para desarrollar un modelo curricular, en que el contenido de un currículo particular puede ser analizado y comparado con otro. Y de Cláudia Bauzer Medeiros, profesora de la Universidad de Campinas y ex presidenta de la Sociedad Brasileña de Computación (2003-2007), quien responde en una entrevista acerca de la enseñanza de esta disciplina. También Patricio Poblete, de la Escuela de Ingeniería y Ciencias de la Universidad de Chile, nos describe el nuevo currículo de Ingeniería adoptado en el último tiempo por la Facultad de Ciencias Físicas y Matemáticas y que es, a grandes rasgos, el marco para el nuevo currículo en Computación. Sobre éste último -que ha sido recientemente adoptado por nuestro Departamento- trata el artículo de Juan Alvarez, Nancy Hitschfeld y José M. Piquer, todos ellos profesores del DCC. Finalmente, tenemos dos artículos en que profesores y colaboradores del DCC nos cuentan acerca de las nuevas tecnologías y metodologías de enseñanza en los cursos de la carrera de Ingeniería Civil en Computación.

Junto con el tema central encontrarán, al igual que en el número anterior, diversas secciones de amplio interés: Investigación Destacada, Computación y Sociedad, Surveys, Grupos de Investigación y Conferencias. Todas ellas esperamos que sean del gusto de ustedes.

Profesor Pablo Barceló
Editor Revista Bits de Ciencia

Estructuras de Datos Compactas

Las diferencias de velocidad en la jerarquía de memoria son cada vez más pronunciadas. Esto ha propiciado una tendencia al estudio de las estructuras de datos que puedan operar usando poco espacio, ya que aunque realicen más operaciones, las pueden ejecutar en una memoria muchas veces más rápida. En una época en que la cantidad de datos a manipular es gigantesca, y las memorias donde pueden ser almacenados son las más lentas de la jerarquía, las estructuras de datos compactas están ofreciendo soluciones muy atractivas en áreas como recuperación de información, Web, bioinformática, bases de datos multimediales, y muchas otras. En este artículo paso revista a algunos de los resultados más espectaculares que ha obtenido esta fascinante área de investigación, e incluyo algunas contribuciones propias relevantes.

INTRODUCCIÓN

Gonzalo Navarro
Profesor Titular, DCC, Universidad de Chile. Doctor en Ciencias mención Computación de la misma Universidad (1998). Sus áreas de interés incluyen algoritmos y estructuras de datos, búsqueda en texto, compresión y búsqueda en espacios métricos.
gnavarro@dcc.uchile.cl

En 1965, el co-fundador de Intel, Gordon Moore, estimaba que "el número de transistores que consigue el menor costo por transistor en un circuito integrado se duplica cada 24 meses". La llamada "Ley de Moore" se ha mantenido válida durante décadas y se aplica a varios aspectos de la tecnología de los computadores: las capacidades de la memoria principal (RAM), el poder de las CPUs, las capacidades de almacenamiento de los discos, etc. Este fabuloso incremento en las capacidades del hardware es en gran parte el responsable de la revolución tecnológica de los últimos 50 años, permitiendo afrontar desafíos de una magnitud impensable hace unos pocos años.

Sin embargo, existen otros aspectos del hardware que no siguen esta ley. Los más notorios son posiblemente la velocidad de acceso al disco y a la RAM. La respuesta a este estancamiento en las velocidades de las memorias ha sido el enriquecer esta jerarquía con nuevas componentes más rápidas, aunque también más caras y (por lo mismo) menores. Las cachés L1, de unos pocos KBs y prácticamente pegadas al chip de la CPU, y luego las L2, de unos pocos MBs, ofrecen "buffers" intermedios, en tamaño y velocidad, entre los registros de la CPU y la RAM de unos pocos GBs. Estos buffers pueden ser desde sólo 2 hasta 50 veces más rápidos que la RAM, y a su vez la RAM es unas cien mil veces más rápida que el disco. Recientemente, las memorias Flash han agregado un nivel entre estas dos, resultando unas 100 veces más rápidas (y menores en capacidad) que los discos. Diversas razones económicas (velocidad vs costo por bit) y físicas (límites fundamentales, distancia a la CPU) hacen pensar que esta jerarquía de memoria se mantendrá por mucho tiempo.

Por otro lado, tenemos las aplicaciones cada vez más ambiciosas en términos de cantidad de datos a generar y manipular. La Web sigue creciendo a un ritmo exorbitante, con cientos de terabytes sólo en la parte estática, y duplicándose aproximadamente cada ocho meses. La minería de su utilización (qué sitios visitan los usuarios, qué consultas hacen) genera un flujo gigantesco de datos diariamente. La bio-informática está cerca de lograr secuenciar cada individuo a un costo razonable, lo cual abre las puertas a tener que mantener los tres mil millones de bases del genoma humano para cada persona. Las aplicaciones astronómicas amenazan con inundarnos de terabytes de datos por día provenientes de los telescopios. La TV y multimedios digitales generan exabytes (millones de terabytes) de material al año. Las tecnologías actuales pueden permitir *almacenar* esta cantidad de datos, pero eso es lo de menos. Lo que se necesita es *manipular* esos datos: analizarlos, transformarlos, extraer información útil de ellos. Y esta tarea es extremadamente penosa si requiere acceder a los datos de forma intensiva y debe llevarse a cabo sobre los datos almacenados en disco.

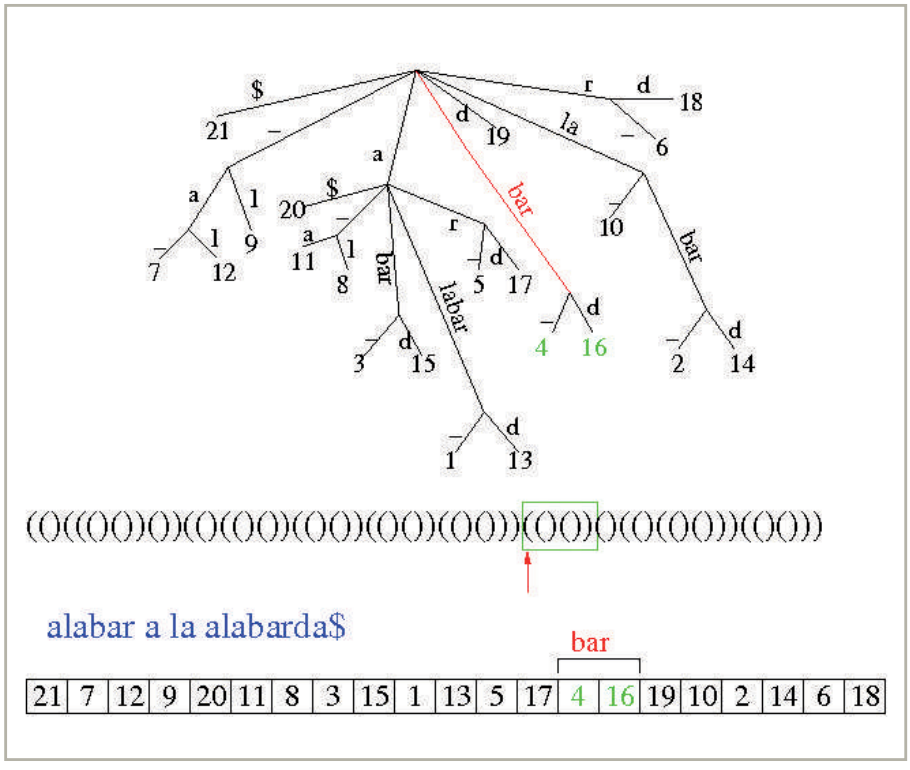
Tomaré un ejemplo bastante modesto como motivación concreta. Supongamos que queremos almacenar un genoma humano (sólo uno, que es lo usual todavía). Este contiene unas tres mil millones de bases, que podemos ver como símbolos sobre un alfabeto de 4 caracteres: A, C, G, T (pueden aparecer otros, pero evitaré complicaciones innecesarias). Almacenando cada carácter en un byte resulta ser cerca de un espacio de 3GB, perfectamente manejable en la RAM de un computador estándar moderno. Más aún, codificando cada carácter en 2 bits podemos almacenarlo en poco más de 700MB.

Sin embargo, los bio-informáticos lo que realmente necesitan es *analizar* este genoma, no sólo almacenarlo. Los tipos de análisis que llevan a cabo consisten, por ejemplo, en buscar secuencias de ADN (es decir, cadenas sobre el alfabeto de cuatro caracteres) para ver dónde ocurren en el genoma, en forma exacta o permitiendo algunas diferencias entre lo que se busca y lo que se encuentra; buscar zonas del genoma que se repiten, en forma exacta o aproximada; descubrir patrones de interés estadístico; etc. Estas búsquedas serían impracticables (por ejemplo, de tiempo cuadrático en el largo del genoma) si no se utilizaran estructuras de datos sobre la secuencia, llamadas *índices*.

El índice más popular por su eficiencia y versatilidad es el *árbol de sufijos*. Éste permite realizar tareas sorprendentemente complejas con un costo de tiempo razonable. Sin embargo, requiere de 10 a 20 bytes por carácter de la secuencia, es decir, unos 30 a 60GB para un genoma humano. Unido a que este tamaño excede en un orden de magnitud lo que hoy en día se puede almacenar en una RAM estándar, se da la desafortunada situación de que los algoritmos sobre árboles de sufijos no son particularmente locales en su acceso a los datos, lo que se paga muy caro cuando se accede al disco. Esta combinación inutiliza, en la práctica, lo que en principio es una excelente solución algorítmica.

A lo largo del artículo mostraré cómo las estructuras de datos compactas permiten representar el árbol de sufijos en poco espacio más del necesario para almacenar la secuencia misma, ofreciendo una solución completamente practicable en espacio y tiempo. En el camino pasaré revista a distintas estructuras de datos compactas necesarias para la solución, y a varias contribuciones propias recientes.

La investigación en estructuras de datos compactas tiene dos componentes: uno es el algorítmico, como es de esperar. El otro, de teoría de la información. Representar los datos en el menor espacio posible obliga a considerar cuánta información hay en ellos y cómo puede representarse en forma compacta. Sin embargo, no se trata sólo de compresión, donde basta justamente con *representar* los datos. Esta vez se necesita



considerar el costo de *acceder* a ellos *sin descomprimirlos*. Esta estrecha interacción entre algoritmos y teoría de la información es una de las cosas más fascinantes de esta área.

EL ÁRBOL DE SUFIJOS

Considere un texto $T[1,n]$ sobre un alfabeto Σ de tamaño σ. Este texto contiene n sufijos: $T[i,n]$ para cada i. Cada sufijo se puede identificar por su posición de partida, i. Consideremos que el último carácter de T es especial, $T[n] = \$$, menor lexicográficamente que todos los otros. Un *arreglo de sufijos* es un arreglo de enteros $A[1,n]$, donde los sufijos se listan en *orden lexicográfico*. Un *árbol de sufijos* es (1) un árbol cuyas hojas son las celdas del arreglo de sufijos, en el mismo orden, (2) tiene las aristas rotuladas con cadenas de caracteres, y (3) la concatenación de los rótulos de la raíz hasta cada hoja es el sufijo correspondiente (no completo, sino expandido hasta el punto en que ningún otro sufijo empieza igual). El árbol no tiene nodos unarios (con un sólo hijo), de haberlos se deben unir en una sola arista. La figura muestra un árbol (arriba) y arreglo de sufijos (abajo) para el texto *"alabar a la alabarda"*. Ignore por ahora los paréntesis que se interponen entre ambos y lo destacado en color.

La utilidad del arreglo de sufijos reside en que todo substring del texto es el prefijo de algún sufijo. Por lo tanto, encontrar todas las ocurrencias de un cierto patrón $P[1,m]$ en T equivale a encontrar todos los sufijos que comienzan con P. Estos forman un rango lexicográfico, por lo cual aparecen todos en un intervalo contiguo del arreglo de sufijos (ejemplificado en la figura con la búsqueda de "bar"). Este se puede determinar mediante dos búsquedas binarias, en tiempo $O(m \log n)$, y luego todas las posiciones del patrón están listadas en el intervalo. Este tiempo es muy bueno, pero el arreglo de sufijos consta de n enteros, que requieren n log n bits , mientras que el texto sólo requiere $n \log \sigma$ bits[1], lo cual es considerablemente menor.

El árbol de sufijos permite realizar esta búsqueda en tiempo $O(m)$, mediante bajar en el árbol y reportar las hojas del subárbol. Si bien la cantidad de nodos en el árbol de sufijos es $O(n)$, y por lo tanto requiere $O(n \log n)$ bits, la constante es de tres a cinco veces la del arreglo de sufijos en la práctica. A cambio, el árbol de sufijos permite hacer otras cosas difíciles o imposibles de simular eficientemente en el arreglo de sufijos. Por ejemplo, para encontrar la auto-repetición más larga en T, basta con recorrer las hojas y reportar la más profunda (en términos del largo de la cadena que representa), lo que resuelve en tiempo lineal algo que requeriría tiempo cuadrático sin el árbol. Otras operaciones muy propias del árbol son el "suffix link", que lleva del nodo que representa la cadena aX al que representa X, donde a es un carácter; y el "ancestro común más bajo" entre dos nodos u y v.

¿CÓMO REPRESENTAR UN ÁRBOL?

Un elemento a comprimir del árbol de sufijos es la topología misma del árbol. Hago notar que hay soluciones donde se prescinde completamente del árbol, pero lo retendré por su valor pedagógico.

Un árbol general con n nodos se representa usualmente con $O(n)$ punteros, los cuales al menos necesitan $O(n \log n)$ bits para que cada puntero pueda distinguir entre n nodos distintos a apuntar. Sin embargo, es fácil ver que esto es un grosero exceso, al menos en términos de la Teoría de la Información. La cantidad de árboles generales distintos de n nodos es el número de Catalán $Cn\text{-}1 = \text{comb}(2n\text{-}2,n\text{-}1)/n = \Theta(4^n/n^{3/2})$, por lo cual deberían bastar $\log C_{n-1} = 2n - O(\log n)$ bits para representar cualquier árbol de n nodos[2].

No es especialmente difícil representar un árbol general usando $2n$ bits. Por ejemplo, podemos recorrer el árbol en pre-orden, abriendo un paréntesis cada vez que bajamos por una arista y cerrándolo cuando subimos. El resultado para el árbol de la figura se dibuja debajo del mismo. Esta representación permite reconstruir la topología del árbol original. ¡Pero eso no es lo que buscamos! Lo que necesitamos es poder navegar en esta representación de paréntesis, tal como navegamos un árbol con punteros.

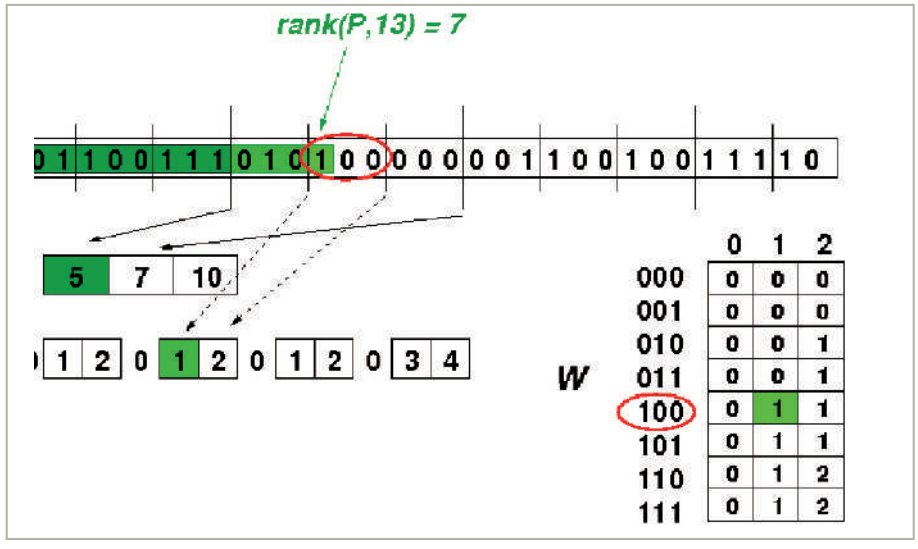
Consideremos que un nodo se identificará con la posición de su paréntesis abierto en la secuencia. Lo primero que quisiéramos sería conocer cuál es su posición en un recorrido en pre-orden (por ejemplo, para almacenar información asociada al nodo en otro arreglo). Esto no es más que la cantidad de paréntesis abiertos hasta el de ese nodo. Si representamos la secuencia de paréntesis en un bitmap $P[1,2n]$, usando el 1 para el paréntesis que abre y 0 para el que cierra, la operación se llama:

$preorden(i) = rank1(P,i)$ = número de 1s en $P[1,i]$.

Asimismo, nótese que la profundidad del nodo i es la cantidad de paréntesis que abren menos la cantidad que cierran hasta la posición i. Esto se llama:

$profundidad(i) = rank_1(P,i) - rank_0(P,i) = 2\, rank_1(P,i) - i$.

¿Cómo podemos calcular rank en tiempo constante, sin utilizar demasiado espacio? La solución es simple y práctica. Consideraremos sólo $rank_1$, ya que $rank_0(P,i) = i - rank_1(P,i)$. Dividimos el bitmap en súperbloques de largo s y bloques de largo b (donde b divide



[1] Los logaritmos serán en base 2 en este artículo.

[2] Dada la pobreza del editor que he sido obligado a usar para este articulo, o mi pobre conocimiento de él, estoy utilizando comb(n,m) para denotar el número combinatorio $n!/(m!(n-m)!)$.

a *s*). Pre-calculamos una tabla *S*[1,*n/s*], donde S[j] = $rank_1$(P,j*s) es el valor de rank1 hasta el comienzo de cada súper-bloque. Pre-calculamos otra tabla *B*[1,*n/b*], donde *B*[*k*] = $rank_1$(*P*,*k***b*) – *S*[floor(*k***b*/*s*)] es el valor de $rank_1$ hasta el comienzo de cada bloque, pero relativo al comienzo de su súper-bloque. Entonces, si *i* = *j***s* + *r* = *k***b* + *r'*, donde *0* <= *r* < *s* y 0 <= *r'* < *b*, significa que *i* pertenece al súper-bloque *s*, bloque *b*, y tiene offset *r'* dentro de *b*. Entonces podemos calcular en tiempo constante

$rank_1(P,i) = S[j] + B[k] + rank_1(P[k*b+1..k*b+b],r'),$

donde el último valor se calcula recorriendo *P* directamente en tiempo *O(b/w)*, siendo *w* la cantidad de bits en la palabra de máquina, mediante técnicas de "popcount". La siguiente figura ilustra el proceso, con s=*9*, *b=3*, y una tabla *W* para procesar *rank* en los bloques en tiempo constante.

Para calcular el espacio necesario se debe considerar que *S* necesita *n/s*log n* bits, *B* necesita *n/b*log s* bits (pues no se representan números mayores que s), y w >= log *n*, la suposición normal en el modelo RAM si se entiende que el computador puede direccionar un número entre *n*. Definiendo *s* = $\log^2 n$ y *b* = log *n*, tenemos tiempo constante y espacio extra O(*n* log log *n* / log *n*) = *o*(*n*) bits.

Algunas operaciones de navegación sobre el árbol son muy simples. Por ejemplo, el primer hijo del nodo *i* es *i+1*, a menos que sea un paréntesis que cierra, en cuyo caso *i* es una hoja sin hijos. Otras operaciones son más complejas. Muchas se construyen sobre dos primitivas fundamentales:

findclose(i) = la posición del paréntesis que cierra *a i*,

enclose(*i*) = la posición del paréntesis que abre más cercano a *i* y que contiene a *i*.

Con estas primitivas, algunas operaciones son sencillas. Por ejemplo, el siguiente hermano de *i* es *findclose(i)+1* (si es que abre. Pues si cierra significa que *i* es el último hijo de su padre), el padre de *i* es *enclose(i)*, el tamaño del sub-árbol de *i* es *(findclose(i) – i + 1) / 2*, etc.

Es imposible detallar la solución a todas las operaciones en este artículo. Para dar una idea, explicaré superficialmente la forma de resolver *findclose(i)*. Se divide la secuencia en bloques de tamaño *b*, y se clasifican los paréntesis que abren como *cercanos* si cierran en el mismo bloque en que abren, y lejanos si no. Los *lejanos* se subdividen en *pioneros*, si el paréntesis lejano previo no cierra en su mismo bloque. Con esta definición resulta que (1) existen *O(n/b)* pioneros (pues conectando los bloques de los pioneros con el de sus parejas resulta un grafo planar), (2) si *i* no es pionero entonces *findclose(i)* está en el mismo bloque del pionero *j* que precede a *i*, y se puede encontrar en tiempo *O(b/w)* recorriendo desde *findclose(j)* con técnicas tipo popcount. Basta entonces marcar los pioneros en un bitmap y almacenar sus respuestas explícitamente, para poder responder *findclose(i)* en tiempo constante si se elige *b* = *O*(log *n*). Esto, sin embargo, todavía requiere de *O(n)* bits extra para almacenar las respuestas a los pioneros. En cambio, se utilizan dos niveles de recursión formando una nueva secuencia de paréntesis pioneros y sus parejas, y volviendo a aplicar la técnica sobre ella. Sólo en el segundo nivel se almacenan explícitamente las respuestas para los pioneros de los pioneros, con lo que se logra el espacio extra *o(n)* y tiempo aún constante.

El bitmap donde se marcan los pioneros, por otro lado, se puede comprimir porque contiene solamente *O(n / log n)* 1s, ocupando espacio *o(n)* y ofreciendo acceso y *rank* en tiempo constante. Esto se consigue, por ejemplo, cortándolo en bloques de largo *b* = (log *n*)/2 y codificando cada bloque como un par *(c,o)*, donde *c* (la *clase*) es la cantidad de 1s del bloque, y *o* (el *offset*) es el índice de ese bloque en una lista de todos los bloques de esa clase (estas listas deben almacenarse explícitamente, pero en total tienen sólo O(sqrt(*n*)) elementos). Los bloques con pocos o muchos 1s tienen clases pequeñas y por ello se necesitan pocos bits para su componente *o*. Con esta técnica, un bitmap *B* con *m* 1s se codifica en $n*H_0(B) + o(n)$ bits, donde $H_0(B) = (m/n) \log n/m + O(m/n)$ es la entropía de orden *cero* de *B*. A esto se le suman otros o*(n)* bits para direccionar la secuencia de campos o y para calcular *rank*.

¿CÓMO REPRESENTAR UN ARREGLO DE SUFIJOS?

Una técnica muy relevante para representar un arreglo de sufijos se basa en la Transformación de Burrows-Wheeler (BWT). Esta produce una permutación *T'* del texto *T*, que se forma escribiendo el carácter que precede a cada sufijo, en el orden en que se listan en el arreglo de sufijos. Resulta que *T'* es más fácil de comprimir que *T* porque contiene largas zonas con el mismo carácter o pocos caracteres distintos (por ejemplo, todas las "C" que preceden a "hile" en un texto probablemente aparecerán seguidas en *T'*), y por ello la BWT se usa en softwares de compresión tan eficientes como *bzip2*.

Ahora bien, la BWT tiene otra propiedad muy relevante para la búsqueda de patrones, que viene de su conexión con el arreglo de sufijos. Digamos que *A* es el arreglo de sufijos de *T*, y que *A[sp,ep]* es el rango de todos los sufijos que comienzan con P. Entonces el rango de sufijos que comienza con *aP*, siendo *a* un carácter, es *A[sp',ep']*, donde

$sp' = C[a] + rank_a(T',sp-1)+1$ y $ep' = C[a] + rank_a(T',ep)$.

Aquí *C* es un arreglo que almacena, para cada carácter, la cantidad de ocurrencias en *T* de caracteres menores a él, y $rank_a(S,i)$ es la cantidad de ocurrencias de *a* en S[1,i]. La razón de esta fórmula es que C[*a*] nos ubica en la zona de *A* de los sufijos que empiezan con *a*. De ellos, los que siguen con *P* son aquellos de *A[sp,ep]* que están precedidos del carácter *a*, es decir, *T'[i] = a, para sp <= i <= ep*, mientras que los que están precedidos de *a* pero son anteriores a *P* son los que cumplen *T'[i] = a* para *i<sp*. Similarmente, si *A[i]=j*, es decir la posición *i* del arreglo de sufijos apunta a la posición *j* del texto, entonces la posición de A que apunta a la posición previa del texto, j-1, es

$LF(i) = C[T'[i]] + rankT'[i](T',i)$.

Con esta herramienta nos basta poder calcular $rank_a$ sobre la secuencia *T'* en tiempo *O(t)* para buscar *P*[1,*m*] en *T* en tiempo *O(t*m)*. Así comenzamos con *sp* = 1 y *ep* = *n*, y procesamos los caracteres de *P* uno a uno hacia atrás. Asimismo, nos basta hacer un muestreo regular en *T*, anotando desde dónde se apuntan estas posiciones muestreadas en el arreglo de sufijos, para poder recuperar los caracteres del texto hacia atrás desde la posición muestreada usando *LF* (una búsqueda binaria en *C* nos dice en la zona de qué carácter de *A* estamos en cada paso). Para esto necesitamos poder acceder a cualquier *T'*[*i*] y, nuevamente, poder calcular $rank_a$ sobre *T'*, y tendremos la capacidad de recuperar cualquier sub-cadena de *T*. Un

mecanismo similar nos permite calcular un valor $A[i]$ sin almacenar A.

El desafío que nos queda, entonces, es representar T' sin usar mucho espacio, y de forma de poder acceder a cualquier carácter y calcular $rank_a$ eficientemente. Con esto seremos capaces de eliminar no sólo A sino también T mismo, y reemplazar todo por T'.

Una estructura extremadamente elegante que permite hacer esto es el *wavelet tree*. Este es un árbol binario balanceado que representará la secuencia S[1,*n*] dividiendo el alfabeto: el árbol tiene σ hojas y log σ niveles. La raíz representa la secuencia completa. Su hijo izquierdo representa la sub-secuencia de S formada por los caracteres de la primera mitad, y el hijo derecho, la sub-secuencia formada por los caracteres de la segunda mitad del alfabeto. Recursivamente, los hijos del segundo nivel dividen el alfabeto en cuatro partes y así sucesivamente.

Cada nodo v del wavelet tree representa entonces una sub-secuencia de S. Sin embargo, ésta no se almacena, sino solamente un bitmap B_v que indica, para cada posición, si el carácter correspondiente pertenece a su hijo izquierdo o derecho. Es fácil ver que el total de bits almacenados en cada nivel siempre suma n, y por lo tanto el total de bits que ocupa el wavelet tree es n log σ, lo mismo que si se representara S en forma plana. Pronto hablaremos de su compresibilidad.

Supongamos que queremos averiguar $S[i]$ a partir del *wavelet tree* de S. Sea v el nodo raíz del *wavelet tree*. *Si B[i]=0*, significa que $S[i]$ pertenece a la primera mitad del alfabeto, y que por lo tanto el carácter está representado en el hijo izquierdo de v. La posición donde está representado depende de cuántos ceros hay en B_v hasta la posición i. Es decir, la búsqueda debe continuar por el hijo izquierdo y debemos reescribir $i = rank_0(B_v,i)$. Similarmente, si $B_v[i]=1$, debemos ir al hijo derecho con $i = rank_1(Bv,i)$. Continuamos este proceso hasta llegar a una hoja, la cual corresponde precisamente al carácter $S[i]$. El tiempo total de obtener el carácter es entonces $O(\log \sigma)$.

El procedimiento para calcular $rank_a(S,i)$ es parecido, sólo que la decisión sobre por cuál rama bajar se toma considerando el carácter a y no el valor $B_v[i]$. Cuando se llega a la hoja correspondiente al carácter a, el valor de i es precisamente $rank_a(S,i)$. La siguiente figura ilustra el proceso para $rank_a(S,16) = 5$, donde S es precisamente la BWT de *"alabar a la alabarda$"*.

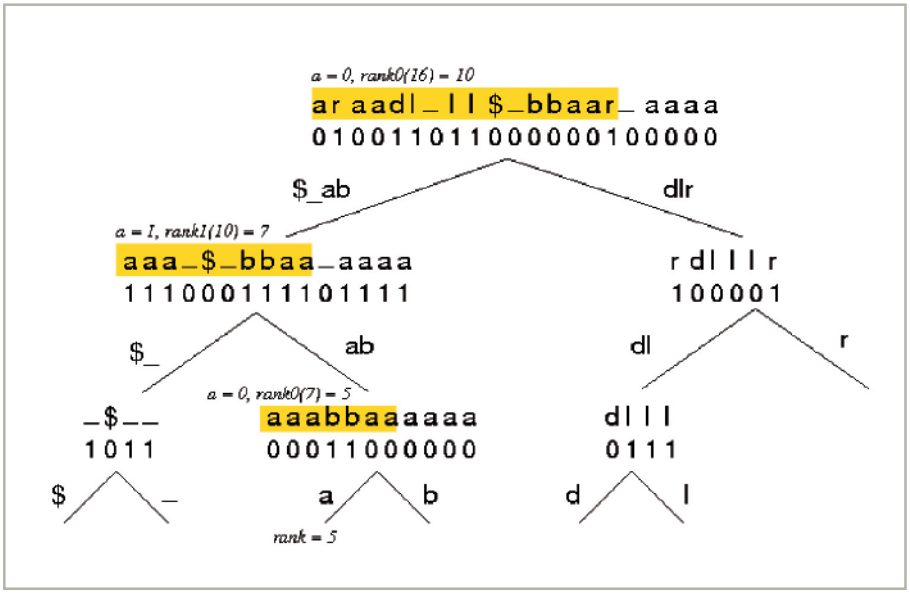
Más aún, si usamos para los bitmaps la representación comprimida de clases y offsets que describimos, resulta que la suma de los $|B_v| H_0(Bv) + o(|B_v|)$ sobre todos los bitmaps del *wavelet tree* totaliza $nH_0(S) + o(n \log \sigma)$, donde $H_0(S)$ es la entropía de orden cero de S, $nH_0(S) = \Sigma_a n_a \log n/n_a$, donde na es la cantidad de ocurrencias del carácter a en S.

La entropía de orden cero es una cota inferior a la cantidad de bits por carácter que puede emitir un compresor estadístico de orden cero, es decir, aquél que modela la frecuencia de cada carácter independientemente de los demás. Nótese que la entropía de orden cero de T' y de T es la misma, por ser permutaciones uno del otro. De este modo se puede obtener un resultado sorprendente. (Las fórmulas donde aparece ε están relacionadas con el período de muestreo mencionado anteriormente.)

Es posible representar un texto T[1,n] sobre un alfabeto σ usando $nH_0(T)+o(n \log \sigma)$ bits, es decir obteniendo compresión, de modo que se puede extraer cualquier substring de largo l del texto en tiempo $O(\log^{1+\varepsilon} n + l \log \sigma)$, contar la cantidad de ocurrencias de un patrón P[1,m] en tiempo $O(m \log \sigma)$, y reportar la posición en el texto de cualquiera de esas ocurrencias en tiempo $O(\log^{1+\varepsilon} n)$, para cualquier constante $\varepsilon > 0$.

Obtuvimos este resultado en 2004 con Paolo Ferragina y Giovanni Manzini, de la Universidad de Pisa, Italia, y con Veli Mäkinen, de la Universidad de Helsinki, Finlandia. La técnica de buscar usando la BWT ya existía (había sido inventada por los coautores italianos en 2000), y el índice se llamaba FM-index. Un problema que tenía era, precisamente, la forma de comprimir S. Los *wavelet trees* también existían desde 2003. Nuestra contribución fue notar que los *wavelet trees* se adaptaban maravillosamente bien al problema que tenía el FM-index, y con ello convertimos al FM-index por fin en una estructura competitiva en la práctica. Publicamos el artículo en *SPIRE* ese año [1] y, en 2007, con algunas mejoras, en *ACM Transactions on Algorithms* [2], una de las dos mejores revistas de algoritmos. Con Paolo Ferragina, su estudiante Rossano Venturini y mi alumno de doctorado Rodrigo González, creamos el sitio *PizzaChili*, en http://pizzachili.dcc.uchile.cl y http://pizzachili.di.unipi.it, donde hay colecciones de texto para experimentar y varios índices de este tipo implementados para su uso libre en la investigación, docencia o industria. Este FM-index es, efectivamente, uno de los más competitivos. Publicamos los resultados prácticos en *ACM Journal of Experimental Algorithms* [3].

Algo un tanto insatisfactorio del resultado enunciado es que la compresión de orden cero es bastante pobre en muchos casos. Un compresor que considera los k caracteres anteriores como contexto para modelar la frecuencia puede obtener mucha mejor compresión (por ejemplo la técnica PPM, usada en compresores como *ppmdi*, posiblemente el compresor de propósito general más efectivo). La cota inferior para



estos compresores es la entropía de orden k, $H_k(S)$, cuya definición más bien técnica omito.

En el artículo mencionado obteníamos en realidad compresión de orden k, mediante particionar el *wavelet tree* en varios, de acuerdo a los contextos de largo k apuntados por el arreglo de sufijos. La técnica requería un tiempo y espacio de construcción considerable. Durante años varios grupos trabajamos en distintas técnicas para obtener compresión a orden k, todas ellas bastante complejas.

En 2007 trabajábamos con Veli en una versión dinámica de nuestro FM-index, es decir, que manejara una colección de textos donde se pudiera insertar y eliminar textos. No lográbamos dar con una forma de mantener actualizada la partición en varios *wavelet trees* que nos garantizara compresión a orden k. Tal vez esa frustración nos iluminó para que notáramos lo que nadie parecía haber visto antes:

Cuando se aplica el método de compresión de pares (c,o) a los bitmaps del wavelet tree, y este wavelet tree representa la BWT de un texto T[1,n] sobre un alfabeto de tamaño σ, el espacio resulta ser automáticamente $nH_k(T) + o(n \log \sigma)$, *para cualquier* $k <= \alpha * \log_\sigma n$ *y constante* $0 < \alpha < 1$.

Esto significa que no había que hacer nada para lograr compresión de orden k, una verdadera ironía después de tanto esfuerzo. Puede ser curioso para el lector el que los experimentos no nos indicaran que el índice se comportaba mejor de lo que nosotros creíamos. La verdad es que no tuvo la oportunidad: nunca habíamos implementado la técnica de los pares *(c,o)*, sino que habíamos optado por usar bitmaps descomprimidos y darle forma de árbol de Huffman al *wavelet tree*, lo cual se veía mucho más práctico y obtenía compresión de orden cero (pero no más que eso). El episodio per se es toda una lección de teoría versus práctica. Luego de este descubrimiento, mi alumno de magíster Francisco Claude implementó la idea y la incorporó a PizzaChili, donde efectivamente resulta ser en muchos casos la mejor alternativa, comprimiendo mucho más que las otras y obteniendo tiempos competitivos.

Publicamos el resultado teórico con Veli en *SPIRE 2007* [4], y luego en 2008 en *ACM Transactions on Algorithms* [5], como parte de un trabajo mucho mayor sobre manejo de colecciones comprimidas y estructuras de datos compactas dinámicas en general. El resultado es el mejor arreglo de sufijos comprimidos existente en términos teóricos, y en cierto sentido cerró la investigación en este tema (si bien hay muchas ramificaciones en las que se sigue trabajando, como memoria secundaria, búsquedas complejas, otros modelos de compresibilidad, etc.). Este trabajo sobre dinamismo, combinado con una mejora posterior obtenida con Rodrigo (publicado en *LATIN 2008* [6] y la versión de revista por aparecer en *Theoretical Computer Science* [7]), también parece haber cerrado el tema del manejo de colecciones dinámicas. Este resultado también permite algo importante: construir el índice utilizando espacio cercano al de la estructura final. Previamente se necesitaba construir la estructura descomprimida para luego comprimirla, lo que plantea un evidente problema de practicidad. En la actualidad mi alumno de magíster Fernando Krell trabaja en implementar esta variante dinámica.

EL ÁRBOL DE SUFIJOS COMPRIMIDO

Parece un juego de palabras astuto decir que, teniendo un arreglo de sufijos comprimido y un árbol comprimido, se tiene un árbol de sufijos comprimido. No está tan lejos de la verdad. Todo nodo del árbol de sufijos se corresponde con un intervalo del arreglo, y no es difícil hacer el mapeo mediante la determinación de cuántas hojas contiene el sub-árbol del nodo. El mapeo inverso requiere determinar el ancestro común más bajo (LCA), que se mencionó anteriormente entre las operaciones que se pueden resolver en árboles comprimidos. Más precisamente, el nodo correspondiente a un intervalo es el LCA de las hojas pertenecientes a las dos puntas del intervalo del arreglo de sufijos. Las cadenas que rotulan las aristas también se pueden obtener: la cadena común entre el primer y último sufijo del intervalo de un nodo v, que el arreglo de sufijos comprimido permite extraer, es la concatenación de los rótulos de las aristas desde la raíz hasta v.

Estas ideas pueden extenderse a una solución que ocupa los $2n$ bits del árbol más el espacio del arreglo de sufijos comprimido, y simula casi todas las operaciones del árbol de sufijos en tiempo constante. Algo insatisfactorio, sin embargo, es que el arreglo de sufijos *determina completamente el árbol*, por lo cual esos $2n$ bits son, estrictamente hablando, redundantes. Esta redundancia se nota mucho cuando el alfabeto es pequeño, como en el caso del ADN. A continuación expondré una solución que prácticamente elimina esa redundancia, al costo de tener tiempos de operación polilogarítmicos.

La idea es no almacenar los nodos del árbol de sufijos, sino trabajar únicamente con intervalos del arreglo. Sin embargo, para poder resolver las operaciones se necesita muestrear algunos nodos del árbol de sufijos, digamos $O(n/s)$, y representarlos explícitamente. Este muestreo debe garantizar que si tomo s veces sucesivamente el suffix-link a partir de un nodo, pasaré necesariamente por un nodo muestreado. No es obvio cómo conseguir ese muestreo, pero tampoco demasiado difícil. Representaremos el árbol de los nodos muestreados usando paréntesis, y utilizaremos también un bitmap que nos permitirá determinar el ancestro muestreado común más bajo (LCSA) entre dos hojas (posiciones del arreglo de sufijos). La siguiente fórmula, válida para todo i, es clave en la solución:

$sdep(LCA(v,v')) >= i + sdep(LCSA(slink^i(v_l), slink^i(v'_l)))$,

donde $sdep(u)$ es el largo del string que lleva de la raíz al nodo u, $slink^i$ es la aplicación del suffix link i veces, v^l es la hoja izquierda del nodo v, y $v'r$ es la hoja derecha del nodo v'. Debido a la condición del muestreo, debe valer la igualdad para algún i entre 0 y s-1, y esto permite detectar cuál es el nodo muestreado en el camino de suffix links. Para estos nodos muestreados guardamos información explícita como $sdep$, y luego podemos devolvernos de los suffix links ejecutados, obteniendo la información de sdep para el nodo original. La fórmula sirve tanto para obtener $LCA(v,v')$ como para obtener $sdep(v) = sdep(LCA(v,v))$, y también $slink(v) = LCA(slink(v_l), slink(v_r))$.

Lo que se necesita para poder aplicar la fórmula es poder calcular el suffix link de una hoja, o de una posición del arreglo de sufijos. Esta es otra hoja, es decir otra posición del arreglo de sufijos: Si $A[i]=j$, el suffix link de i es la posición i' que apunta a j+1. Nótese que esto es precisamente la inversa de la función LF(i). Esta se puede calcular con la fórmula

$LF^{-1}(i) = select_a(T', i-C[a])$,

donde *a* es el carácter con el que empiezan los sufijos de la zona del arreglo de sufijos a la que *i* pertenece, y $select_a(S,j)$ es la inversa de $rank_a$: la posición de la *j*-ésima ocurrencia de *a* en *S*. Se calcula usando el *wavelet tree* de *S* en tiempo $O(\log \sigma)$, recorriéndolo desde la hoja de *a* hacia la raíz, y ejecutando select sobre los bitmaps del *wavelet tree*. El *select* sobre bits es más complejo que el *rank*, pero también se puede resolver en tiempo constante y espacio extra sub-lineal.

Una vez que estas tres operaciones claves sobre intervalos del arreglo de sufijos están resueltas, varias otras se pueden construir sobre ellas o directamente usando el arreglo de sufijos. Por ejemplo, el padre de un nodo v es el menor entre los intervalos $LCA(v_l\text{-}1,v_l)$ y $LCA(v_r,v_r+1)$, mientras que el *i*-ésimo carácter de la cadena representada por un nodo *v* es el que corresponde a la zona del arreglo de sufijos donde cae $LF^{-i}(v_l)$. Esta inversa iterada se puede calcular mediante $LF^{-i}(j) = A^{-1}[A[j]+i]$. Tanto *A* como A^{-1} se calculan con los muestreos del texto almacenados en el arreglo de sufijos comprimido.

Eligiendo adecuadamente *s*, podemos tener un árbol de sufijos donde el espacio extra sobre el del arreglo de sufijos es despreciable, y que soporta una amplia gama de operaciones en tiempo $o(\log^2 n)$. Publicamos este trabajo con Luís Russo y su director Arlindo Oliveira, de la Universidad Técnica de Lisboa, Portugal, en *LATIN* 2008 [8], y la versión dinámica en *CPM* 2008 [9]. La implementación muestra que con un 10% de espacio extra sobre el arreglo de sufijos comprimido se obtienen tiempos de operación razonables (entre centésimas y cienmilésimas de segundo, dependiendo de la operación). Para el genoma humano el espacio total sería aproximadamente 1.6 GB (incluyendo la secuencia). Paralelamente, estoy trabajando con mi alumno de magíster Rodrigo Cánovas en implementar en forma práctica otra propuesta publicada con Veli y Johannes Fischer [10], donde los tiempos son sub-logarítmicos y el espacio es algo peor, si bien aún depende de la entropía de orden k.

CONCLUSIONES Y PERSPECTIVAS

En este artículo he querido dar al lector un panorama del tipo de trabajo que se hace en estructuras de datos compactas, mediante la elección de un problema paradigmático y el recorrido por la investigación que llevó a resolverlo. Este camino pasa por varios problemas fundamentales y elegantes de estructuras de datos, y por algunas contribuciones propias. He evitado, para no abrumar al lector, las referencias a cada resultado mencionado, limitándome a las atingentes a mi propia contribución (¡que era lo que al fin y al cabo se me había pedido exponer!). Para quien quiera saber más recomiendo un survey reciente [11].

Las estructuras de datos compactas resultan un área tremendamente atractiva desde varios puntos de vista. A quienes aman la belleza de los algoritmos y las estructuras de datos, les ofrece un campo nuevo donde hay que reinventar soluciones para problemas que están completamente resueltos en el ámbito clásico. A quienes se sienten fascinados por la teoría de la información y la compresión, les brinda una combinación exquisita y sutil con las estructuras de datos, donde el problema trasciende al de meramente representar la información, obligando a investigar tanto entre oportunidades de compresión como de eficiencia de acceso que no existen en la compresión clásica. E invitando a profundizar en la dualidad entre compresión e indexación, ambas basadas de algún modo en extraer las regularidades de los datos. A quienes disfrutan con las soluciones prácticas a desafíos formidables, les ofrece una herramienta fresca y poderosa para hacer frente a la masividad de la información típica de nuestros tiempos. Y a quienes toda la vida hemos tratado de caminar por el estrecho desfiladero entre la teoría y la práctica, intentando no caer en la teoría inútil ni en la práctica vacía de fundamento científico, nos da un maravilloso ejemplo de cómo ambas pueden hacer milagros cuando trabajan juntas.

Para finalizar doy las gracias a mis ex-alumnos Francisco Claude y Rodrigo Paredes, quienes aportaron con valiosas sugerencias. BITS

REFERENCIAS

[1] Paolo Ferragina, Giovanni Manzini, Veli Mäkinen, and Gonzalo Navarro. An Alphabet-Friendly FM-index. *Proc. SPIRE'04*, pages 150-160. LNCS 3246.

[2] Paolo Ferragina, Giovani Manzini, Veli Mäkinen, and Gonzalo Navarro. Compressed Representations of Sequences and Full-Text Indexes. *ACM Transactions on Algorithms (TALG)* 3(2), article 20, 24 pages, 2007.

[3] Paolo Ferragina, Rodrigo González, Gonzalo Navarro, and Rossano Venturini. Compressed Text Indexes: From Theory to Practice. *ACM Journal of Experimental Algorithmics (JEA)* 13:article 12, 2009. 30 pages.

[4] Veli Mäkinen and Gonzalo Navarro. Implicit Compression Boosting with Applications to Self-Indexing. *Proc. SPIRE'07*, pages 214-226. LNCS 4726.

[5] Veli Mäkinen and Gonzalo Navarro. Dynamic Entropy-Compressed Sequences and Full-Text Indexes. *ACM Transactions on Algorithms (TALG)* 4(3):article 32, 2008. 38 pages.

[6] Rodrigo González and Gonzalo Navarro. Improved Dynamic Rank-Select Entropy-Bound Structures. *Proc. LATIN'08*, pages 374-386. LNCS 4967.

[7] Rodrigo González and Gonzalo Navarro. Rank/Select on Dynamic Compressed Sequences and Applications. To appear in *Theoretical Computer Science*.

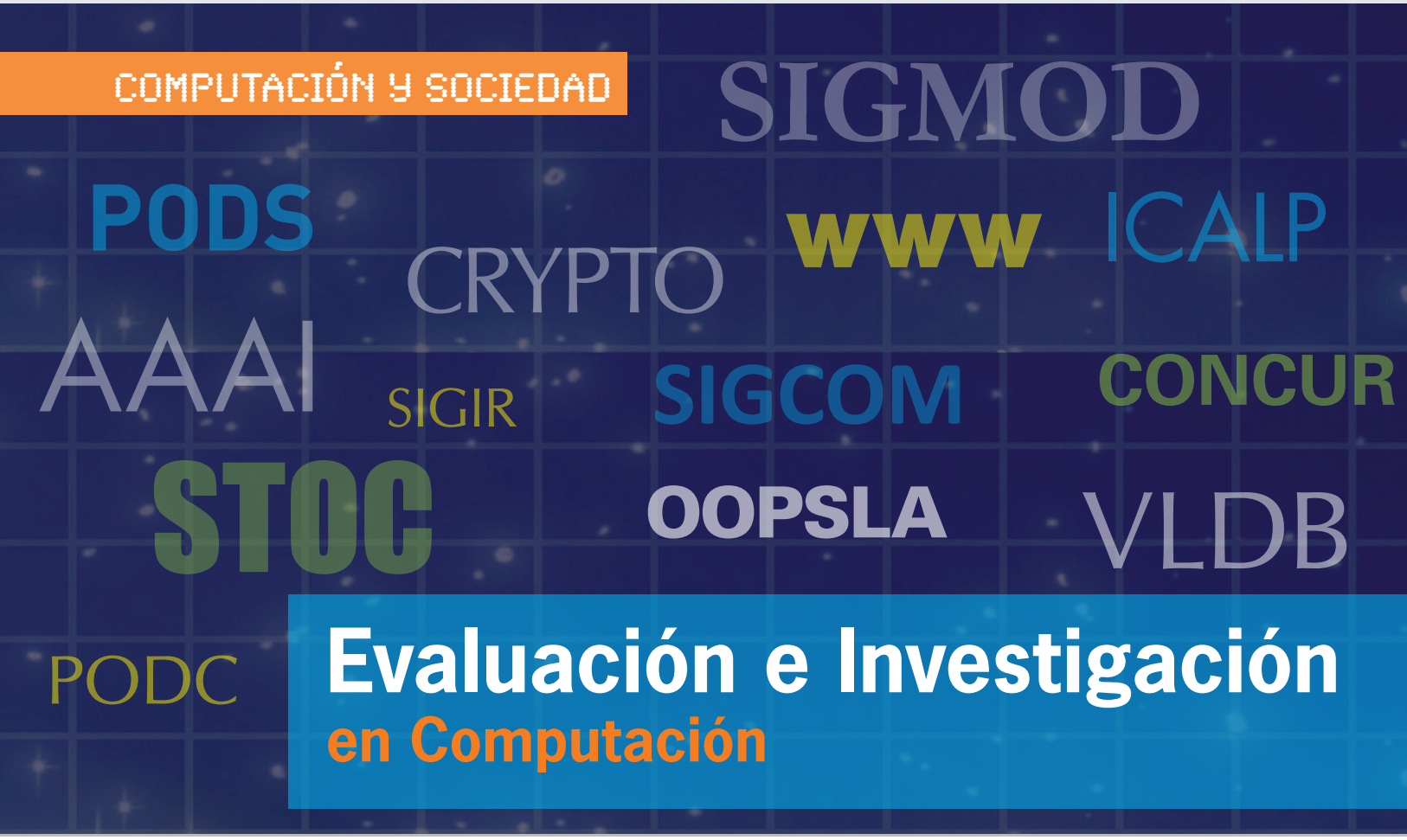
[8] Luís Russo, Gonzalo Navarro, and Arlindo Oliveira. Fully-Compressed Suffix Trees. *Proc. LATIN'08*, pages 362-373. LNCS 4957.

[9] Luís Russo, Gonzalo Navarro, and Arlindo Oliveira. Dynamic Fully-Compressed Suffix Trees. *Proc. CPM'08*, pages 191-203. LNCS 5029.

[10] Johannes Fischer, Veli Mäkinen, and Gonzalo Navarro. An(other) Entropy-Bounded Compressed Suffix Tree. *Proc. CPM'08*, pages 152-165. LNCS 5029.

[11] Gonzalo Navarro and Veli Mäkinen. Compressed Full-Text Indexes. *ACM Computing Surveys* 39(1), article 2, 61 pages, 2007.

COMPUTACIÓN Y SOCIEDAD



Evaluación e Investigación en Computación

Claudio Gutiérrez
Profesor Asociado, DCC, Universidad de Chile. Ph.D. en Computer Science de Wesleyan University, Estados Unidos. Investigador Asociado del Centro de Investigación de la Web y el Grupo Khipu de bases de datos.
cgutierr@dcc.uchile.cl

Los procesos de evaluación son parte esencial del mundo moderno. ¿Cómo evaluar en la disciplina conocida como Computación o Informática? La noción de "evaluar" puede tener un sentido bastante amplio: evaluar un profesional, un profesor, un producto, un software, una investigación. Puesto de otra forma: ¿Cuál es el perfil de un buen profesional de esta área? ¿Qué competencias debe tener? ¿Qué es una buena carrera? Ni que decir sobre ¿qué es un buen software? o más difícil: ¿Qué es una buena investigación? El tema se ha hecho más relevante por la preeminencia que ha ido ganando la disciplina en el ámbito de la producción y la vida diaria (Internet, redes sociales, televisión y telefonía digital con sus chiches asociados, etc.). Puestas las múltiples aristas que emergen, en este artículo queremos concentrarnos en una de ellas: la evaluación de esta disciplina en la academia. ¿Qué es ser un buen profesor de Computación? ¿Qué es hacer buena "ciencia" en Computación? ¿Qué es buena tecnología en Computación?

Para responder a la pregunta sobre la evaluación necesariamente debemos comenzar por contestar ¿Qué es la Computación? Sin saber de qué estamos hablando es difícil determinar qué es bueno. Este es uno de los puntos centrales de un implícito debate en el mundo científico en Chile. Hasta hace poco, se asociaba la Computación a un servicio[1] y quienes se dedicaban a ella eran tales: proveedores de servicios. No tenían un objeto propio como actividad. Hoy esa opinión es insostenible y la Computación se levanta como una disciplina más entre otras [2, 3, 1]. Partamos

intentando determinar cuál es su núcleo, centro, esencia, para determinar lo que todo exponente de esta área debiera saber.

Hace mucho tiempo se pensaba que era el computador. Computer Science le dicen aún a esta disciplina en idioma inglés: Ciencia del computador. Un absurdo que ha llevado a muchos malentendidos. Es como llamar a la Astronomía "Ciencia del Telescopio". Hoy por hoy, está medianamente claro que hay un núcleo de problemas significativos que son el objeto de esta disciplina: los procesos de información y la automatización de dichos procesos. Así las nociones de algoritmo, especificación, lenguaje, codificación, están en el centro de ella. Y por supuesto la forma cómo los humanos por un lado, y las máquinas (el hardware) por el otro, interactuán con ellos. Pero el objeto de nuestra disciplina no es ni el humano ni el hardware por sí mismos, sino el puente entre ellos.

Una primera distinción, absolutamente novel, es que los objetos con que trabajamos no son físicos, sino virtuales. No son átomos sino bits (como decía Negroponte en un brillante ensayo[5]). Esto hace que la disciplina tenga un carácter muy distinto de las ciencias experimentales clásicas, en sus metodologías como en su evaluación. Tanto que a los biólogos, químicos, físicos, etc. les cuesta hasta reconocer que "la Computación" pudiera tener el mismo estatus como disciplina que la de ellos.

Por otra parte, uno podría estar tentado de asociarla entonces con las Matemáticas, una cercanía que también ha producido malos entendidos. Con las Matemáticas tiene metodologías comunes, particularmente los temas de formalismos (ambas comparten estructuras conceptuales, lenguajes formalizados, nociones como demostración, etc.), pero por su objeto de estudio difieren fundamentalmente. Hay algunas áreas de intersección, por ejemplo, combinatoria, grafos, complejidad, que de alguna manera encierran las Matemáticas que son más "útiles" a la Computación y, por el otro lado, los temas de Computación que son más matemáticos. Pero son sólo eso: intersecciones. El enfoque y el objeto de un matemático difiere vastamente del usado en el área de Computación. El objeto matemático no interesa per se en Computación. Interesa en tanto ayuda a capturar el objeto de la Computación.

Una tercera vertiente que complica más aún este espectro de problemas son los temas sociales y humanos asociados a la Computación. En sus comienzos, la disciplina estaba fuertemente cargada hacia el lado de la máquina. Podríamos decir que el puente partió construyéndose desde este lado: hardware, circuitos, lenguaje de máquinas, etc. Poco a poco fue acercándose al humano (quienes programaban en ese tiempo claramente no eran humanos...). Y hoy estamos cargados hacia el otro lado del puente: los temas de interacción con el humano (usabilidad, interfaces, entre otros), aspectos sociales (por nombrar algunos: redes sociales, usos de datos, comportamiento) y los que involucran a ambos como semántica, lenguajes de alto nivel, etc. han ido ganado tal preeminencia que la pregunta de si la Computación es una ciencia natural (es decir, de objetos naturales) o social/humana ha llegado a tener sentido [3].

En este contexto estamos instalados y necesitamos evaluar. Y permítasenos entonces reducirnos –como lo indicamos antes– a la evaluación académica. Dejaremos de lado los aspectos docentes, que merecen un artículo por sí mismo. Sólo para ilustrar la densidad del tema, he aquí algunos tópicos que ese artículo debiera abordar. ¿Qué es un buen docente en el área? ¿Un teórico? ¿Alguien que domine la práctica de la Computación? ¿Es necesario que el profesor sepa recorrer el puente de extremo a extremo, que conozca la gama de tópicos a los que pueden orientarse los alumnos de Computación? ¿Cuáles son los temas indispensables de un currículum de Computación? ¿Cómo debe estar organizado? ¿Cuáles son las carreras "naturales" a que conducen los estudios en esta disciplina? Dónde debiera "vivir" una carrera de Computación: ¿entre las ingenierías? ¿las ciencias? ¿de manera independiente? Son algunas de las muchas preguntas que un artículo tal debiera responder.

Vamos a lo nuestro. Crecientemente una parte importante de la evaluación en la academia se refiere a la producción científica. Desde antiguo, en las áreas tradicionales, ésta se refiere fundamentalmente a comunicaciones científicas ("artículos", mejor conocidos hoy por su nombre inglés *papers*, que se ha transformado en sinónimo de "artículo científico"). No trataremos aquí otra línea de productos científicos conocidos como patentes (que por sus aristas ligadas a la producción, como a la compleja noción de "propiedad" en el área del software y los

Una primera distinción, absolutamente novel, es que los objetos con que trabajamos no son físicos, sino virtuales. No son átomos sino bits (como decía Negroponte en un brillante ensayo[5]). Esto hace que la disciplina tenga un carácter muy distinto de las ciencias experimentales clásicas, en sus metodologías como en su evaluación.

1 "Ofimática" le decían en CONICYT hasta hace poco; muchos siguen creyendo que se trata sólo de servicios.

datos, merece también otro artículo por sí mismo). El gran tema es entonces cómo se evalúa un *paper*.

En las ciencias "clásicas" la evaluación se hace mediante un proceso conocido como "referato entre pares" (traducción del peer reviewing en inglés). Esto ha sido implementado en esas disciplinas a través de revistas especializadas (journals en inglés), que tienen un comité editorial consistente en un grupo de especialistas (de tamanño variable según la revista, pero usualmente de un número pequeño de dos cifras). Estos, cuando algún autor (puede ser un individuo o un grupo) presenta un artículo, eligen dos o tres revisores que evalúan la calidad de la investigación presentada, la pertinencia (a la revista), la novedad, etc. y el jefe editorial y su equipo deciden en base a esa evaluación si el artículo es aceptable o no. Hay una tercera opción, bastante común, en que se le pide al autor que mejore ciertos aspectos, que cambie otros, etc., como condición para ser aceptado.

Este proceso ha cristalizado ciertas prácticas que nos gustaría destacar:

(1) El proceso es de "ventanilla abierta", es decir, en cualquier momento el autor puede someter su trabajo a evaluación.

(2) El conjunto de editores (quien decide al final qué es bueno o malo) es relativamente fijo en el tiempo (digamos, no cambia en lo sustantivo de año en año por ejemplo).

(3) El proceso se hace desde diferentes lugares geográficos. Usualmente hay muy poco contacto "físico" entre revisores y editores y ninguno entre autores.

(4) En muy pocos casos hay fallos definitivos, es decir, existe la posibilidad de interactuar y discutir los fallos, y presentar más evidencia si es necesario.

(5) El tiempo de revisión no está fijado de antemano (aunque crecientemente este elemento está siendo incorporado a las revistas).

Por una razón histórica y sociológica (que no está bien estudiada aún), la comunidad de Computación siguió otro camino. En sus orígenes, los precursores más "científicos", que estaban bastante ligados a las Matemáticas o a la Electrónica, cuando publicaban artículos lo hacían en journals de corte matemático o de eléctrica, o crearon algunos bastante similares (e.g. CACM en 1958, JACM en 1959). Otros lo hacían en journals de sistemas como el IBM System Journal (1961). Pero aquí es crucial señalar que en los comienzos de la Computación, muchos de sus héroes desarrollaban productos, sea hardware, software o algo intermedio. Estos eran sus productos estrellas y, adicionalmente, no como lo central, publicaban un reporte técnico sobre ello o exhibían resultados en algún seminario. Así fueron presentados lenguajes como FORTRAN y ALGOL y muchos otros, y así se presentó la teoría de compiladores, pilares centrales de la disciplina. Poco más tarde todo lo asociado al fantástico proyecto de Small Talk, y hasta más reciente, las ideas de creadores como Douglas Engelbart o Tim Berners-Lee, y otros, han estado esencialmente ausentes de los journals en su sentido clásico.

Estos comienzos dificultosos, donde mucha de la actividad se hacía al margen de la noción de journal (al contrario, por ejemplo, de las Matemáticas y la Física), hicieron que la gente del área creara un tipo de evento muy particular, las conferencias, cuyo objeto era reunirse a presentar sus trabajos e interactuar. La noción no era nueva. Desde siempre, desde la creación misma de las sociedades científicas hace varios siglos, el evento central de éstas era la reunión. Históricamente, los miembros se reunían a "presentar" (leer) sus trabajos, los que posteriormente eran publicados en los Proceedings (actas) de la Sociedad. Nótese que la evaluación no se hacía al trabajo, sino a la persona: quien (lograba) ingresar a ese selecto Club, ya tenía un estándar y, por defecto, todo lo que produjera era bueno. Para una sociedad donde los científicos eran una elite pequeña funcionaba. Pero claramente este método no escala. Esta dificultad –incompatible con las necesidades

científicas de la revolución industrial– hizo que con el tiempo este proceso único se dividiera en dos: por un lado, los journals de hoy, donde cada científico puede enviar sus trabajos para evaluación (de hecho muchos mantienen aún la marca de este origen, por ejemplo Transactions of the Royal Society). Y, por otro, las reuniones (generalmente anuales) de la Sociedad respectiva, que tienen un carácter social, donde cada miembro presenta sus trabajos, teniendo éstos carácter de "comunicación" informal, es decir, no son evaluados previa presentación.

Bueno, en nuestra disciplina, probablemente siguiendo alguna de esas ideas y adaptándolas a métodos y "productos" (el software) no conocidos antes, se produjo una creación monstruosa: las conferencias. Las conferencias nacieron como una suerte de reuniones científicas (como las de las sociedades científicas antiguas), pero democráticas, esto es, donde todos podían participar y ser evaluados. Este nuevo "frankenstein", a quien la comunidad científica clásica aún no logra entender ni le ha dado carta de ciudadanía, se convirtió a poco andar en el principal medio de evaluación científica en el área. Expliquemos un poco más el evento "Conferencia". Estos son eventos periódicos (generalmente anuales o bianuales), de algún grupo de especialización dentro de la Computación, cuyo objeto principal es la presentación de trabajos científicos. ¿Cómo lograr que no se transforme en una reunión de miles, dispareja, donde junto a un gran trabajo pudiera presentar un charlatán? Esto supuestamente no podía ocurrir en las reuniones de las sociedades científicas clásicas, pues la membresía a la sociedad estaba restringida. Digamos de paso que este carácter elitista –los mismos miembros "elegían" a sus nuevos miembros, o algunas veces había que esperar a que muriera uno pues el número era fijo– limitaba de manera severa el carácter necesariamente amplio que iba tomando la ciencia. Las Conferencias logran el milagro de abrir a todos las reuniones científicas y, a la vez, mantener la rigurosidad científica por medio de un método extraordinariamente ingenioso: El Comité de Programa, que es una suerte de Comité Editorial (de las revistas), que evaluá por peer reviewing los artículos y productos (software, demostraciones, etc.) que se envían. Esta simple idea logró introducir muchas novedades en el proceso de evaluación de la investigación científica.

Comparémosla con la lista anterior que hicimos para las revistas:

(1) El proceso es periódico y con deadline, es decir, hay plazos fijos para someter un trabajo a evaluación.

(2) El Comité de Programa (conjunto de editores que deciden finalmente qué es bueno o malo) cambia completamente de año en año.

(3) Aunque el proceso de evaluación se hace desde diferentes lugares geográficos, uno de los objetivos centrales del evento es el contacto "físico" entre autores.

(4) Hay fallos definitivos, es decir, no hay posibilidad de discutirlos y presentar más evidencia si es necesario.

(5) El tiempo de revisión está fijado de antemano (por los tiempos que existen entre el período de envío, revisión y realización del evento).

¿Cómo se compara este proceso con el de los journals? ¿Qué ventajas y desventajas tiene? Dejamos al lector que medite sobre ello. La intención de este artículo es sólo presentar las particularidades históricas y sociológicas que ha tomado el proceso de evaluación de la investigación en el área de Computación, e intentar que un público más amplio comprenda las vicisitudes de este proceso en la disciplina. Aquel ha sido muy mal entendido por la comunidad científica en general, y chilena en particular, asumiendo desde un principio que no "corresponde" a las tradiciones científicas establecidas, y luego que sus productos no tendrían la misma validez que tienen aquellos de los journals clásicos. Todo esto, contradiciendo abiertamente los hechos, que indican que esta disciplina es la que más avances ha tenido en las últimas décadas y de alguna manera ha guiado los avances tecnológicos que conforman gran parte del mundo actual.

Recientemente este proceso ha estado en las primeras planas de la discusión académica en el área. Para el lector interesado que quiera seguir el debate, recomendamos los últimos números de Communication of the ACM y los diversos punteros que de allí se derivan. Un buen punto de partida es [4].BITS

REFERENCIAS

[1] Robert L. Constable, Computer Science: Achievements and Challenges circa 2000, Cornell University, March 2000.

[2] Peter Denning, Is Computer Science Science?, CACM, April 2005. (Hay versión castellana).

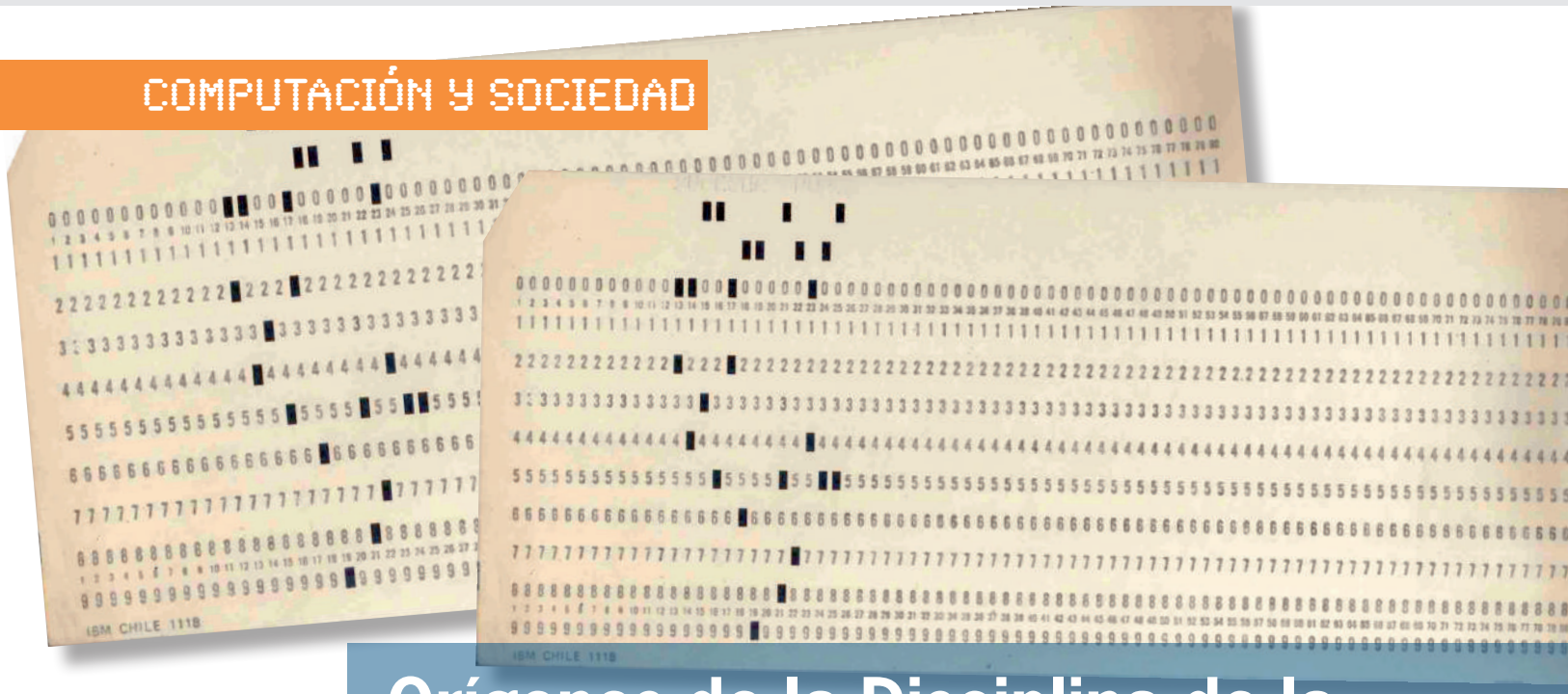
[3] Peter Denning, Computing is a Natural Science. ACM Communications, July 2007.

[4] Peter J. Denning, Paul S. Rosenbloom, Computing: The Fourth Great Domain of Science, CACM, September 2009.

[5] Nicholas Negroponte, Being Digital, Vintage Publ., 1995. (Hay traducción castellana: Ser Digital).

[6] Moshe Vardi, Conference vs. Journals in Computing Research, Comm. ACM, Mayo 2009, Vol. 52, N°5.

COMPUTACIÓN Y SOCIEDAD



Orígenes de la Disciplina de la Computación en Chile 1961-1975

(Resumen Extendido)

Juan Alvarez
Académico DCC, Universidad de Chile. Master of Mathematics (Computer Science), University of Waterloo. Ingeniero de Ejecución en Procesamiento de la Información, Universidad de Chile.
jalvarez@dcc.uchile.cl

Claudio Gutiérrez
Profesor Asociado, DCC, Universidad de Chile. Ph.D. en Computer Science, Wesleyan University, Estados Unidos. Investigador Asociado del Centro de Investigación de la Web y el Grupo Khipu de bases de datos.
cgutierr@dcc.uchile.cl

1. INTRODUCCIÓN

Los primeros computadores digitales llegaron a Chile hace casi medio siglo. Desde esa época se acumuló experiencia que gatilló el surgimiento posterior de una disciplina científica y de ingeniería, que ha madurado lo suficiente como para comenzar a hacer un balance y preocuparse de su pasado y orígenes. Cabe señalar que para designar la disciplina, en Chile se usan casi indistintamente los términos "computación" e "informática", provenientes de la traducción de su designación en lengua inglesa y francesa respectivamente. Nosotros usaremos aquí "computación", lo que no debe entenderse como toma de posición respecto del nombre o carácter de la disciplina.

En este contexto se planteó la idea de estudiar la historia de la Computación en Chile con el propósito de desentrañar el desarrollo del área en el país. La evolución de la disciplina, como otras de carácter científico e ingenieril, ha sido influenciada por las grandes tendencias internacionales. Sin embargo, presenta importantes e interesantes particularidades locales. Para investigar este desarrollo estamos trabajando en el proyecto "Orígenes de la Disciplina de la Computación en Chile 1961-1975", que ha involucrado investigación documental, entrevistas, recuperación de material gráfico y escrito. Creemos que este estudio y reflexión sobre la historia de la Computación en Chile ayudará a consolidar la identidad y la comunidad de la disciplina, documentar experiencias y a formar una base para proyectar su futuro en el país.

Una de las características del desarrollo inicial de la Computación en Chile es la presencia bastante compartimentada de tres actores: las universidades, el Estado y los privados. Hemos considerado el período 1961-1975 cuyos delimitadores son, por un lado, el ingreso al país del primer computador digital y, por el otro, los comienzos del reconocimiento académico de la disciplina. En este período la demanda por la "Computación" provino fundamentalmente del Estado y las respuestas fueron de parte del mismo Estado y las universidades. Respecto de los privados, algunas empresas

Conmemoración de los 40 años de ECOM, diciembre de 2008: Isaquino Benadof, Raimundo Beca, Italo Bossi, Hugo Segovia.

nacionales comienzan a incorporarse como usuarios. Por su parte, las grandes compañías extranjeras fabricantes de computadores juegan un rol de proveedores, de apoyo técnico lateral y de puente con los productos (equipos) del nor-atlántico.

En lo que sigue presentamos un resumen de esa investigación, poniendo énfasis sobre los principales procesos y actores institucionales de los primeros años de la disciplina de la Computación en Chile: el Estado y las universidades. Al respecto, en este resumen hemos evitado mencionar individuos y sus roles (siempre complejos y controversiales, particularmente cuando la distancia histórica es cercana), para centrarnos en los procesos, demandas, actores institucionales y aspectos técnicos (sin detallar las máquinas y los lenguajes utilizados).

2. LA COMPUTACIÓN EN EL ESTADO

El Estado chileno acumuló experiencia en la automatización del procesamiento de datos en algunos servicios desde muy temprano. La primera iniciativa se remonta al censo de 1930, con el uso de tarjetas perforadas. Estas primeras prácticas se fueron ampliando progresivamente para realizar procesamiento más elaborado de tarjetas (clasificación, tabulación, etc.) a través de máquinas UR (Unit Record).

Posteriormente, a comienzos de la década del 60, varias instituciones del Estado adquieren computadores digitales para enfrentar los desafíos impuestos por el creciente volumen de datos y garantizar mayor rapidez, confiabilidad y flexibilidad en su procesamiento. En 1961, el servicio de Aduanas de Valparaíso instala el primer computador de orientación administrativa: un IBM-1401 (comercializado por IBM a partir de 1959). En seguida se incorporan computadores similares a otros organismos (Tesorería, Ferrocarriles, Impuestos Internos, Capredena, FACH, etc.).

Por otra parte, entre las empresas fiscales productivas, la compañía de Aceros del Pacífico (CAP) en Talcahuano (1963) instala un IBM-1401 con el primer tambor magnético para almacenamiento secundario. En 1966 ENAP adquiere un IBM-360, el primero de su tipo en Chile, con tecnología de "tercera generación" (circuitos integrados) que produce la obsolescencia de los transistores. En 1974 la misma empresa compra un IBM-370 que destina principalmente a aplicaciones del área técnica.

Estos primeros computadores se utilizaron sobre todo en aplicaciones administrativas o de gestión ("comerciales" en la nomenclatura de la época), que involucraban muchos datos pero con cálculos aritméticos sencillos (sumas, promedios, porcentajes, etc). Su programación se realizaba principalmente en el lenguaje COBOL (COmmon Bussiness Oriented Language) y de forma secundaria en RPG (RePort Generator).

Considerando que laborar en Computación se consideraba muy estresante, las jornadas de trabajo no podían superar las 6 horas diarias. Además de los iniciados en el "arte" de programar, se contaba con analistas de sistemas (que diseñaban los sistemas computacionales), operadores ("de consola" y "periféricos") y digitadoras (que traspasaban información desde papel a tarjetas perforadas o medios magnéticos). La capacitación en un principio corrió por cuenta de los fabricantes de computadores, y posteriormente por la empresa estatal de Computación.

Cabe señalar que existía el espacio de la "programación de sistemas", esto es el desarrollo de programas "utilitarios" y aplicaciones complementarias a los sistemas operativos destinados a facilitar el desarrollo del resto de las aplicaciones. Su programación se realizaba en el lenguaje Assembler (notación simbólica del lenguaje binario de las máquinas) complementado con "macros" (instrucciones simbólicas que se traducían por varias instrucciones de Assembler). De esta manera se explotaban eficientemente los escasos recursos de memoria y CPU de los computadores de entonces, pero con un significativo mayor costo de programación, debido a la complejidad de la "idiosincrasia" de las máquinas.

La Empresa Nacional de Computación (ECOM)

La necesidad de centralizar esfuerzos y reducir costos, especialmente de los carísimos computadores de entonces, culmina con la creación en 1968 de la empresa nacional y estatal de Computación (ECOM) adscrita a la CORFO. Esta iniciativa se enmarca en los esfuerzos de planificación estatal y complementa la creación de otros organismos orientados a fomentar la tecnología como INTEC.

ECOM se consolidó rápidamente y jugó un rol muy activo. Por una parte, impartió cursos de capacitación de programación y análisis de sistemas. Por otra, desarrolló sistemas computacionales genéricos para resolver los problemas de procesamiento de datos más frecuentes en la industria y los servicios (sistema contable, cuentas corrientes, sueldos, etc). Adicionalmente, en un esfuerzo conjunto con CORFO e INTEC, implementó a comienzos de los años 70 el original e innovador proyecto Synco (o Cybersyn) cuyo propósito fue planificar y

coordinar la producción en las empresas del área de propiedad social.

ECOM adquirió inicialmente equipos IBM-360. En 1972, y con el propósito de diversificar los proveedores, compró un equipo Burroughs 3500 y en 1973 encargó a la CII de Francia un Iris-80 para Santiago y un Iris-60 para la subgerencia Bío-Bío (que en 1975 daría origen al Centro Regional de Computación e Informática de Concepción - CRECIC). A partir de septiembre de 1973 la intervención de ECOM cancela la compra de los computadores CII y comienza a limitar su rol promotor y centralizador de la informática desde el Estado.

Seguidamente, en 1974 se decreta la rebaja temporal (por un par de meses) de los aranceles de importación de computadores. Como resultado se produce un ingreso masivo de máquinas que amplía de manera abrupta la Computación en el sector privado, siendo el incipiente sector financiero el mayor comprador y usuario de tecnología. La nueva situación generó un fuerte déficit de especialistas que tuvo que abordarse con un plan nacional intensivo de capacitación de especialistas (PLANACAP) coordinado por ECOM y la colaboración de las universidades de Chile, Técnica del Estado y Católica de Chile.

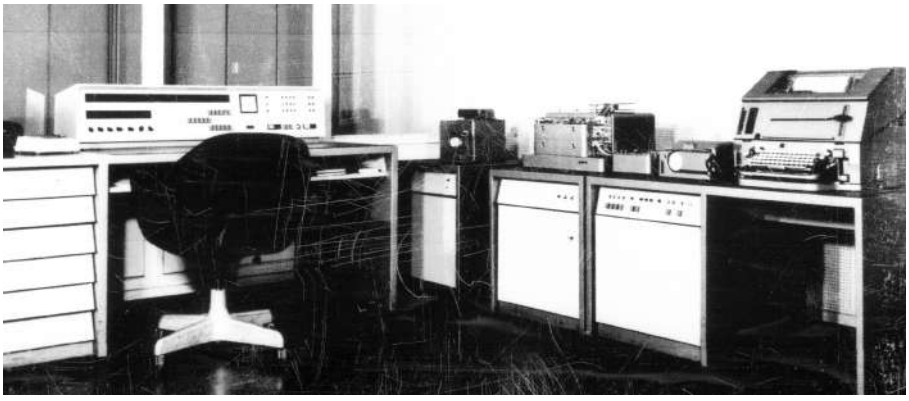
Cabe señalar que, aunque de menor tamaño e importancia, a fines de los sesenta comenzó la "computación administrativa" en el ámbito privado. En 1967 COPEC-Abastible adquiere un computador NCR-315. Por otra parte, algunas empresas se hacen usuarias de computadores universitarios procesando "jobs" o en el modo "block time" (arriendo por horas). Es el caso de la Compañía Manufacturera de Papeles y Cartones (CMPC) desde 1969. Ese mismo año los bancos de Chile, Edwards y Cobrechuqui (de propiedad mixta) adquirieron computadores NCR Century 200. Posteriormente se establecen las primeras empresas consultoras especializadas (por ejemplo, SONDA en 1974).

Wolfgang Riesenkönig y Guillermo González. Instalaron el ER-56 ("Lorenzo") en la Escuela de Ingeniería de la Universidad de Chile en 1962. (Fotografía enero 2009).

3. LA COMPUTACIÓN EN LAS UNIVERSIDADES

El fenómeno de la Computación también atrajo tempranamente la atención de diversos académicos en varias universidades. A fines de los 50, en algunas universidades se desarrollaron experiencias de "computación analógica", proceso que disminuyó su importancia con la llegada de los computadores digitales.

El primer computador digital de orientación científica llega en 1962 a la Escuela de Ingeniería de la Universidad de Chile: se trataba del computador alemán Standard Elektrik Lorenz ER-56 (conocido como "Lorenzo") con tecnología de transistores, memoria para 3000 palabras de 7 dígitos (9K), tambor magnético de 60K y lectora de cinta de papel perforado ("teletipo"). Operaba sin sistema operativo a través de programas escritos en los lenguajes de máquina y ALCOR (ALgol COnverteR). Su costo fue cerca de 200.000 dólares



Computador ER-56 (Fotografía gentileza del profesor Riesenkönig).

Seguidamente, las universidades Católica de Chile (1963), Federico Santa María (1964) y de Concepción (1966) adquieren computadores IBM-1620, comercializados desde 1959 para el ámbito académico y científico. Por su parte, en 1964, la Universidad Técnica del Estado (hoy USACH) recibe como donación un computador Datatron.

En 1966 la Universidad de Chile instaló un IBM-360, computador de "propósito general", cuyo objetivo de diseño era abarcar los 360º del espectro de aplicaciones: científicas, administrativas y de sistemas. La máquina, que funcionaba con los sistemas operativos OS o DOS, tenía 128K de memoria, 7.2M en disco, una unidad de cinta magnética, una lectora de tarjetas perforadas y una impresora para formularios continuos de papel. Un par de años después, la Universidad Católica adquiere un computador Burroughs-3500.

El propósito principal de los primeros computadores fue apoyar los cálculos necesarios para resolver problemas de otras disciplinas: cálculo numérico, ecuaciones diferenciales, estadística, programación lineal, cálculo estructural, análisis de redes, etc. Los programas se escribían principalmente en el lenguaje FORTRAN (FORmula TRANslator) y eran diseñados y programados por los propios científicos e

ingenieros, con el apoyo eventual de alumnos y memoristas. Para facilitar la enseñanza se utilizó una versión educacional de FORTRAN diseñada en la Universidad de Waterloo de Canadá: WATFOR (WATerloo FORtran).

Como etapa previa a la programación se utilizaban "diagramas de flujo" para representar gráficamente el orden de ejecución de las instrucciones. Su uso fomentaba la programación en base a las instrucciones de control IF y GOTO, produciendo programas "spaghetti", es decir, desordenados y carentes de estructura. A comienzos de los 70 surge, como reacción, la "programación estructurada" que utilizaba tres estructuras de control: secuencia, selección (if-else) e iteración (while). Algunos investigadores usaron y enseñaron programación científica estructurada en ALGOL (ALGorithmic Language) y programación de sistemas en el lenguaje PL/360 (una versión estructurada del lenguaje de máquina del IBM-360).

La instalación de los primeros computadores en las universidades requirió de la organización de centros de Computación. Su principal función fue administrar el uso y operación de las máquinas y ofrecer servicios de programación a distintos usuarios universitarios y externos. Para satisfacer la creciente demanda, los computadores funcionaban prácticamente las 24 horas del día los 365 días del año. Por otra parte, los centros tuvieron también la responsabilidad de ofrecer los primeros cursos de Computación y Programación para los potenciales usuarios. Esta labor fue complementada con la publicación de apuntes e informativos respecto del uso del hardware y software disponible.

Primeras carreras

La rápida difusión de las capacidades y potencial de los computadores gatillaron la creación de la primera carrera profesional universitaria de Computación: Programación de Computadores, de 3 años de duración. En coincidencia, y en el contexto del proceso de reforma que vivían con el propósito de modernizarse para responder a las necesidades del país, las universidades de Chile (1968), de Concepción (1970) y Católica crean las primeras carreras de Programación de Computadores.

Posteriormente, y en concordancia con las necesidades de preparar profesionales con mayor formación, se crean las carreras de Ingeniería de Ejecución, de 4 años de duración. La Universidad de Chile transforma en 1971 su carrera de Programación en Ingeniería de Ejecución en Procesamiento de la Información, logrando titular a sus primeros egresados en 1973. La Universidad Técnica del Estado crea en 1972 la Ingeniería de Ejecución en Computación e Informática. Y en 1975 la Universidad Técnica Federico Santa María ofrece la Ingeniería de Ejecución en Sistemas de Información.

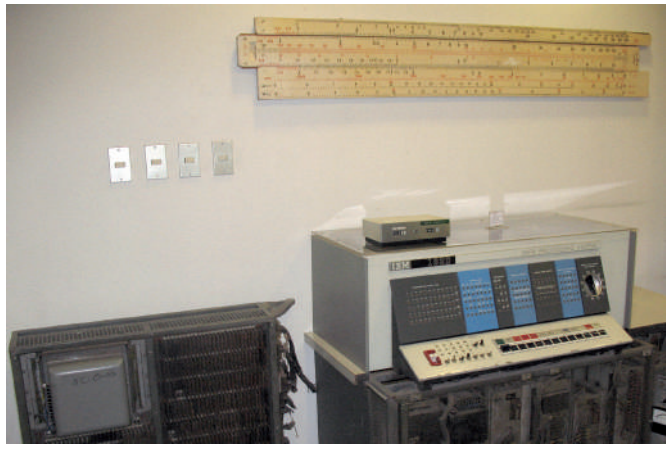
Cabe señalar que años después, coincidiendo con un profundo cambio en la legislación universitaria impuesta en 1981, se crearon las primeras carreras de Ingeniería Civil en Computación e Informática, de 6 años de duración, equiparando el tiempo de formación con las especialidades de Ingeniería más tradicionales.

Departamentos

El desarrollo del área motivó y promovió, no sin oposición de las disciplinas de ingeniería tradicionales, la creación de los primeros departamentos académicos de la disciplina, con las funciones universitarias de docencia, investigación y extensión en las tres dimensiones de la disciplina: ciencia, tecnología e ingeniería.

En 1975 se crean coincidentemente departamentos académicos de Computación en tres universidades. En la Universidad Técnica Federico Santa María surge el Departamento de Ingeniería Informática que ofrece un Posgrado en Ciencia de la Computación e Informática. En la Universidad Técnica del Estado se crea el Departamento de Matemática y Ciencia de la Computación, impartiendo la Licenciatura en Matemática y Ciencia de la Computación, con la orientación principal de formar profesores. En la Universidad de Chile se organiza el Departamento de Ciencias de la Computación, ofreciendo un Magíster en Ciencias mención Computación.

Si bien la creación de los departamentos científicos universitarios consolida formalmente la disciplina, la actividad de investigación había comenzado previamente, tanto en los centros de Computación, como en los distintos departamentos académicos en que residían los investigadores del área (matemáticas, electricidad, industrias). Prueba de ello es la realización anterior de congresos especializados. En 1968, en la Universidad Técnica Federico Santa María se realiza el "Primer Encuentro Nacional de Computación" organizado por la ACHITI (Asociación Chilena de Tratamiento de la Información). Posteriormente en 1974, la Universidad Católica de Valparaíso organiza el "I Panel de Discusión sobre Tópicos de Computación". El evento que se planteó inicialmente con una cobertura regional, luego fue nacional y finalmente latinoamericano, dando origen al CLEI (Centro Latinoamericano de Estudios en Informática) cuya conferencia acaba de realizar su edición número 35.



Computador IBM-1620 de la Pontificia U. Católica de Chile, PUC. (Fotografía septiembre 2009, gentileza del Museo del Depto. de Ciencia de la Computación, PUC).

Pablo Fritis, Hugo Segovia, Víctor Sánchez: diseñadores de la primera carrera de Computación en la Universidad de Chile en 1968 (Fotografía enero 2009).

4. EPÍLOGO

En síntesis, en la década del 60 llegaron los primeros computadores digitales, tanto a algunas instituciones del Estado donde se realizó principalmente "computación administrativa", como a varias universidades donde se hizo fundamentalmente "computación científica". Estas vertientes evolucionaron y convergieron, gatillando el surgimiento de la disciplina de la Computación a mediados de la década del 70.

En el Estado la experiencia en el uso de la tecnología en diversos servicios y empresas motivó la creación de la Empresa Nacional de Computación. En el sector universitario, los actores principales fueron las entonces dos universidades estatales nacionales (de Chile y Técnica) y cuatro privadas subvencionadas (Santa María, de Concepción, Católica de Chile y Católica de Valparaíso). Los fabricantes de computadores, principalmente IBM y secundariamente NCR y Burroughs, jugaron un rol que fue más allá de la venta de máquinas, asesorando y capacitando al personal técnico. Resulta destacable señalar también que, tanto en el Estado como en las universidades, hubo tres polos geográficos de desarrollo con importantes relaciones y comunicación entre ellos: Santiago, Valparaíso y Concepción.

Finalmente, detrás de esta evolución se encuentran personas de la industria y la academia, algunos de los cuales han sido informantes de nuestro proyecto. En este breve resumen, por motivos de espacio, no nos hemos referido a ellos. Sin embargo no quisiéramos dejar de agradecer aquí a quienes han contribuido con su tiempo y voluntad de manera invaluable a reconstruir la historia de los primeros años de la Computación en Chile: Alfredo Acle, Raimundo Beca, Isaquino Benadof, Italo Bossi, Víctor Canales, Federico Cavada, Armando Cisternas, Enrique D'Etigny, José Durán, Yussef Farrán, Pablo Fritis, Fernando García, Guillermo González, Juan Hernández, Patricio Maturana, Jaime Michelow, Aldo Migliaro, Waldo Muñoz, Wolfgang Riesenkönig, Oscar Sáez, Víctor Sánchez, Hugo Segovia, Fernando Villanueva, Julio Zúñiga. Nuestro agradecimiento también a Jorge Rozas, quien colaboró en la primera etapa de la investigación. BITS

Fernando Vildósola, Efraín Friedmann, Roberto Frucht, Wolfgang Riesenkönig en el I Encuentro Nacional de Computación en la UTFSM, 1968 (Fotografía gentileza del profesor Riesenkönig).

SE CAYÓ EL SISTEMA

En el marco de las entretenidas conversaciones con algunos de los pioneros del área aparecieron muchas anécdotas, aunque no todas ellas se pueden contar. Entre las que sí, parece que llegamos por fin a conocer el origen de la frecuente y socorrida expresión "se cayó el sistema". Según nuestros informantes, en Chile la expresión tendría dos fuentes explicativas. Por una parte, están los que la atribuyen al hecho que entonces los programas se perforaban en tarjetas, que se amarraban con un elástico (respecto de los elásticos habría mucho paño que cortar especialmente entre los operadores...). Los programas más grandes se mantenían en cajas con capacidad para dos mil tarjetas. En más de una ocasión a nuestros pioneros se les cayó la caja, que contenía el programa del sistema computacional, es decir, se les "cayó el sistema".

Otros de nuestros informantes atribuyen la expresión a una situación mucho más dramática. Cuenta la leyenda que una institución porteña, motivada por la impaciencia de tener pronto su primer computador instalado y funcionando, tuvo la temeraria iniciativa de trasladar e instalar "personalmente" la pesada máquina a uno de los pisos superiores de su edificio. Como los computadores de entonces eran muy voluminosos, se contrató un servicio de grúa, obviamente sin mucha experiencia en computadores, y adivinen qué pasó. ... ¡bingo! se les "cayó el sistema".

Sitios de interés

http://www.laopinon.cl/admin/render/noticia/10505

http://ingenieria.uchile.cl/revista/43/index.html

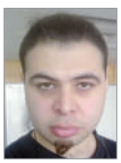
http://www.informatics.indiana.edu/edenm/publications/publications.html

Buscador de la Transparencia

DCC, Universidad de Chile Centro de Investigación de la Web

Senén González
Estudiante de Doctorado en Ciencias de la Computación, DCC, Universidad de Chile. Egresado de Ingeniería Civil en Computación, de la misma Universidad.
sgonzale@dcc.uchile.cl

Mauricio Marín
Investigador del laboratorio Yahoo! Research Latin America, Universidad de Chile. Investigador asociado del Centro de Investigación de la Web. PhD en Computer Science, University of Oxford, UK.
mmarin@yahoo-inc.com



Víctor Sepúlveda
Estudiante de Magíster en Ciencias mención Computación, DCC, Universidad de Chile, bajo la supervisión del profesor Benjamín Bustos. Egresado de Ingeniería Civil en Computación de la misma universidad.
vsepulve@dcc.uchile.cl

1. INTRODUCCIÓN

Durante este año 2009 un equipo de estudiantes y profesores del DCC ha estado participando en el desarrollo del llamado "Buscador de la Transparencia" (http://buscador.chileclic.gob.cl/). Se trata de una herramienta destinada a apoyar la implementación de la nueva Ley de Transparencia Activa promulgada recientemente por el Gobierno de Chile. El objetivo es permitir a cualquier persona buscar información publicada por los distintos sitios Web del Estado y municipalidades.

Los ministerios, municipalidades y otras reparticiones del Estado están obligadas a publicar periódicamente información pertinente a la Ley en forma oportuna. El Buscador proporciona herramientas que permiten a los administradores de los respectivos sitios Web publicar dicha información en un formato estándar, lo cual no sólo abarca el llenado de información vía formularios por parte de los encargados de los sitios Web, sino también la presentación de resultados desde consultas automáticas a sistemas de bases de datos. Dicha estandarización también hace posible supervisar a nivel central el cumplimiento de la normativa de publicación y la actualización de la información de Transparencia.

La primera versión del Buscador entró en operación en mayo de 2009. Desde

entonces ha tenido varias extensiones. Sus inicios no estuvieron exentos de críticas e incomprensión respecto de sus objetivos de diseño y ventajas. La comparación obvia fue con Google, el cual, si bien se puede restringir a dominios específicos, sigue siendo un buscador genérico que al final resulta ser menos efectivo por las siguientes razones:

El Buscador de la Transparencia está centrado en un contexto bien particular, con una problemática asociada a un crecimiento muy dinámico de los sitios Web del Estado. Por ejemplo, cuenta con un sistema diseñado para mantener e incorporar nuevas semillas desde donde hacer nuevas colectas de información. Dichas semillas se pueden agrupar en entidades lógicas que facilitan su administración e indexación. Respecto de las búsquedas puede ser visto como un sistema segmentado. Da la posibilidad de buscar sobre todos los sitios del Estado y municipalidades, considerando por separado tanto las secciones destinadas a publicar información de la Ley de Transparencia Activa como al resto de los contenidos de los distintos sitios Web. Es decir, los resultados de las búsquedas se presentan de manera separada para reducir las interferencias entre ambos tipos de contenidos, y existe separación entre sitios del Estado y municipalidades.

Asimismo, también por construcción, el Buscador permite focalizar las búsquedas solamente dentro de lo que se conoce como ChileClic, "La Guía de Servicios del Estado": un portal administrado por Estrategia Digital (ministerio de Economía) cuyo objetivo principal es entregar información destinada a describir trámites y proporcionar orientación sobre los distintos servicios que el Estado ofrece a chilenos y extranjeros.

Los estudiantes del DCC que han participado en todas las etapas del proyecto de construcción del Buscador de la Transparencia son: Senén González, Víctor Sepúlveda, Eduardo Graells y Mauricio Monsalve. También han participado los profesores Ricardo Baeza-Yates, Claudio Gutiérrez y Mauricio Marín.

A continuación se presenta una descripción general del diseño del Buscador.

2. DESCRIPCIÓN GENERAL

El Buscador de la Transparencia es un sistema que tiene tres componentes principales:

MyWeb: conjunto de herramientas para recolectar sitios, crear índices de búsqueda y realizar consultas sobre esos índices (utiliza tecnología del buscador TodoCL).

Panel de Control: sistema basado en Web, que permite gestionar las herramientas y datos de MyWeb. Realiza operaciones tales como la gestión de URLs que serán recolectados e indexados por el Buscador.

Gestor de Contenidos o CMS: parte pública del Buscador (sitio Web incorporado a ChileClic) que posee un panel de control para gestionar la información que éste publica y la interacción con otros sitios del gobierno.

2.1. MyWeb

MyWeb es considerablemente más rápido y liviano que otros sistemas de dominio público, lo que es una ventaja frente al alto tráfico de consultas. Este era el único componente existente al momento de iniciar la construcción del Buscador. Contiene cuatro componentes:

Recolector (o crawler): que recibe una dirección de inicio, usualmente la página principal de un sitio o dominio. Descarga todas las páginas que encuentra dentro del mismo dominio/sitio que tengan profundidad física mayor o igual a la de la dirección inicial.

- También puede recibir una blacklist de palabras para rechazar ciertas URL.
- Baja páginas HTML y archivos PDF, PPT, XLS, DOC y TXT.

Indexador: crea un índice a partir de una o más colectas realizadas por el crawler. Las colectas pueden ser de sitios distintos, de modo de mezclarlas todas dentro de un único índice. Todos los documentos bajados con éxito por el crawler son indexados.

Demonio de Consultas: componente que está continuamente escuchando en un puerto de la máquina en busca de consultas. Cuando recibe una, busca los documentos relevantes en el índice.

- Los documentos son ordenados por relevancia de texto de acuerdo a las palabras presentes en la consulta.
- Puede responder consultas diferenciando o ignorando acentos.
- AND: busca documentos que sean relevantes para algunas de las palabras de la consulta. Equivalente a buscar +palabra1 +palabra2 en otros buscadores. Los documentos seleccionados contienen todas las palabras de la consulta.
- OR: busca documentos con al menos una de las palabras de una consulta. Equivalente a la búsqueda por omisión en otros buscadores. Los documentos seleccionados pueden contener una o más palabras de la consulta.
- EXACT: busca la frase exacta de las palabras de la consulta. Equivalente a escribir la consulta entre comillas en otros buscadores: ''palabras de consulta''.
- Se pueden correr distintos demonios de manera simultánea, de modo que cada uno responda consultas utilizando distintos índices (los resultados pueden ser mezclados en el CMS).
- Se entregan sugerencias a consultas que tienen pocos o ningún resultado. Estas sugerencias provienen de nuevas consultas que producen una cantidad suficiente de resultados, las cuales son construidas de tal manera que la distancia de edición respecto de la consulta original es pequeña.
- Entrega un máximo de 200 resultados por consulta. Para obtener un mayor número, el usuario puede refinar su consulta incluyendo palabras más

específicas o seguir las sugerencias entregadas por el Buscador.

Procesador de Consultas: componente que recibe la consulta y se comunica con el demonio para obtener los resultados. Este componente se puede modificar para adaptarlo a las necesidades del cliente (el procesador, al igual que los siguientes dos componentes, fue desarrollado específicamente para el Buscador).

2.2. Panel de Control

El Panel de Control se utiliza para gestionar los índices y colectas de MyWeb. Permite crear varias instancias de MyWeb para focalizar el Buscador en distintos segmentos de la Web, tal como ocurre actualmente donde se recolectan e indexan por separado distintos tipos de sitios Web del gobierno (ministerios, municipalidades, etc.). Cuenta con la siguiente funcionalidad:

- Crear/editar/eliminar colectas.
- Especificar semillas para las colectas.
- Agrupar colectas en índices.
- Ver registro (logs) de funcionamiento del Buscador.

El panel es de uso interno. Sólo pueden acceder a él los administradores del sistema mediante una interfaz Web (utiliza PHP y MySQL para administrar la base de datos).

2.3. CMS

El gestor de contenidos tiene dos partes:

Frontend: es el sitio del Buscador, accesible desde http://buscador.chileclic.cl. Recibe las búsquedas ingresadas en el sitio y en las cajas de búsqueda externas y muestra los resultados.

- Permite a usuarios del gobierno registrarse para acceder al backend.
- Permite presentar una nube de consultas más frecuente. En esta nube las consultas se realizan por "frase exacta" (por

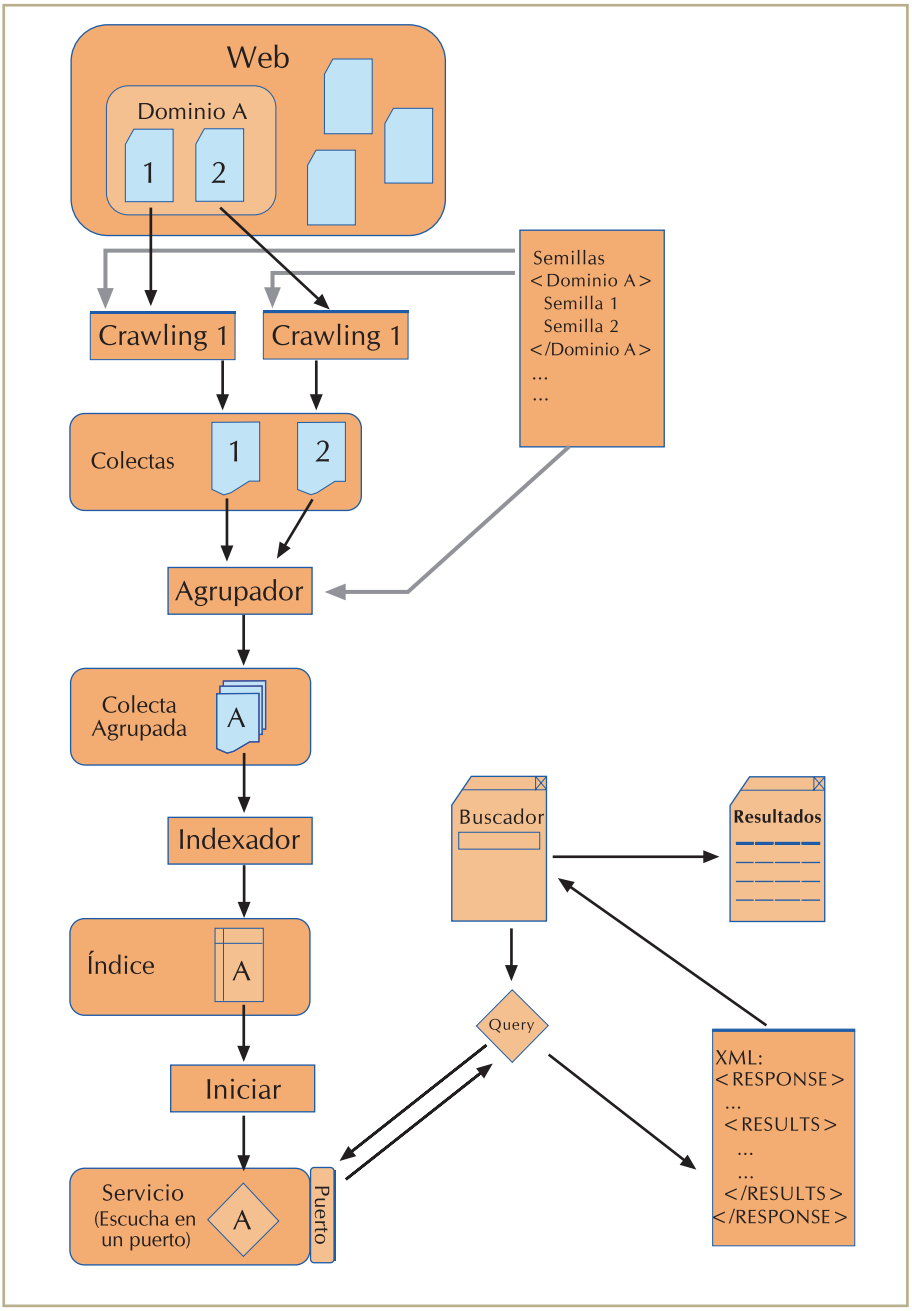


Figura 1: Funciones principales del Buscador.

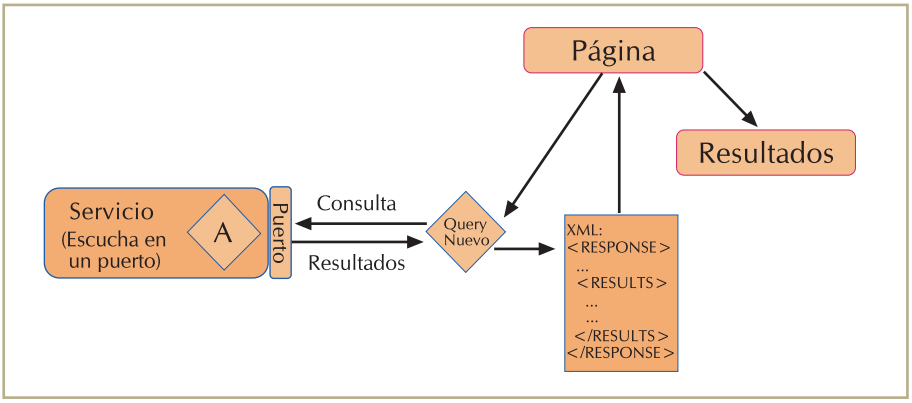


Figura 2: Sistema de respuestas.

ejemplo: "licencia de conducir"). En caso de no encontrar resultados, se utiliza el filtro "todas las palabras".

Backend: es el panel interno del gestor de contenidos. Entre otras cosas, admite:

- Gestionar la presentación y contenidos de la nube de consultas.
- Generar reportes de uso: consultas por tipo y sitio, y cantidad de consultas en los últimos períodos (día/semana/mes).
- Crear cajas de búsqueda que llevan al usuario al sitio del Buscador.
- Gestionar usuarios con acceso a los paneles de gestión de nubes de consulta, reportes de uso y de cajas de búsqueda.
- Crear nubes de consultas para las consultas más frecuentes. Se pueden aplicar filtros de número mínimo de consultas, blacklist de palabras y período de tiempo.

2.4. Detalles Adicionales

El esquema general del funcionamiento del Buscador de la Transparencia se muestra en la Figura 1. El crawler examina y descarga una parte de la Web, los sitios de interés para el sistema, y los almacena para su posterior análisis. Luego el Indexador es activado para revisar las páginas recolectadas y construir con su información índices que serán utilizados durante la posterior fase de consulta. El módulo de consultas consta de un conjunto de sistemas independientes que permanecen activos, respondiendo los requerimientos hechos con los índices que se han construido en la etapa anterior.

Tal como se muestra en la Figura 2, cada sistema de respuesta es un servicio al que se le ha asignado un puerto para que reciba consultas y entregue respuestas. La página utiliza uno de los programas de consulta para comunicarse con dicho puerto, realizar las preguntas ingresadas por el usuario y desplegar los resultados de la forma más adecuada. El sistema de procesamiento de consultas responde con un archivo XML

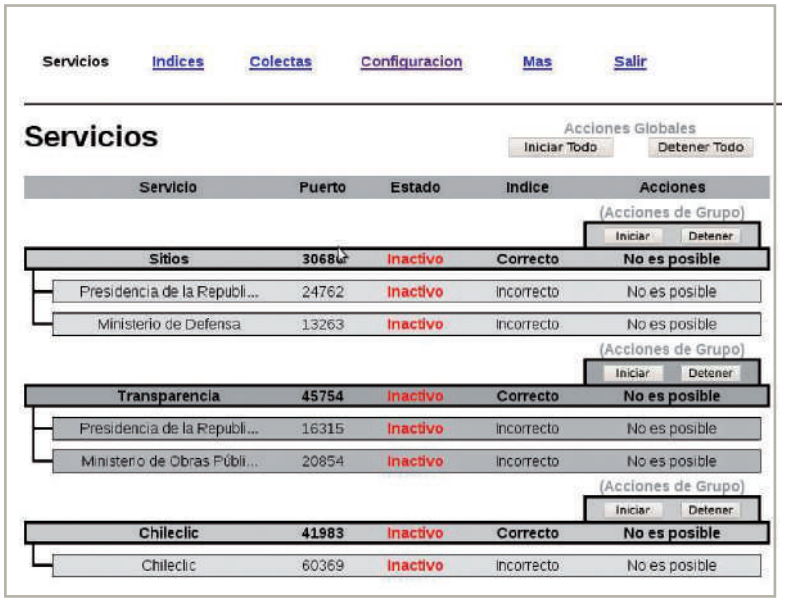


Figura 3: Interfaz de servicios.

Figura 4: Interfaz de Índices.

Figura 5: Mantenedor de colectas (general).

que contiene los resultados de la consulta. Los datos contenidos en los campos XML se utilizan para construir la página Web que se presenta al usuario como respuesta a su consulta.

En la Figura 3 se muestra la interfaz que permite levantar el servicio de respuesta a las consultas, pudiéndose levantar un servicio de respuestas por cada índice creado. Las acciones disponibles son iniciar o detener servicio.

En la Figura 4 se exhibe la interfaz que admite crear los índices a partir de las colectas realizadas. Sólo es posible crear los que han sido definidos en la configuración y que no están vacíos o corruptos. Además es posible borrarlos. También se pueden crear índices globales (que contengan todas las semillas del sistema), e índices de grupo que contienen los índices de algunos de los dominios generados en la interfaz de configuración. En esta Figura se muestra la interfaz que permite controlar las colectas y la Figura 5 describe los detalles de colectas individuales.

Es en la interfaz de configuración donde se configura el sistema añadiendo, borrando o modificando grupos, dominios o semillas. Cada semilla representa un sitio de gobierno que se desea colectar. Estas se agrupan en dominios, por ejemplo ministerio de Salud, el que contendría todas las semillas del Ministerio. Luego los dominios se congregan en grupos generales, los cuales sirven para diferenciar los tipos de dominio que son. Por ejemplo, si son sitios Web de organismos gubernamentales, de Transparencia o si son de municipalidades. Pueden existir muchas clasificaciones. Por cada grupo o dominio se genera un índice. El dominio contiene el índice de todas sus semillas y el de grupo contienen un índice de todas las semillas de sus dominios. La interfaz que permite realizar la gestión de semillas se muestra en la Figura 7.

Dentro de la configuración del sistema existen los enlaces; asociaciones de dominios entre los distintos grupos. De esta manera se pueden asociar los sitios de los organismos con sus páginas de Transparencia. La interfaz se muestra en la Figura 8. BITS

Figura 6: Mantenedor de colectas (detalles).

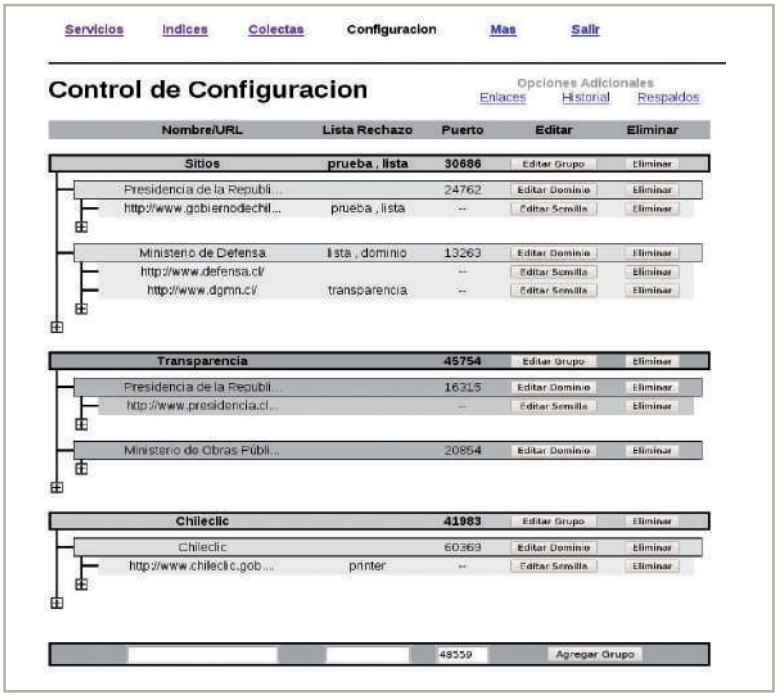


Figura 7: Interfaz de configuración.

Figura 8: Interfaz de configuración (enlaces).

EDUCACIÓN EN INFORMÁTICA

Computing Curricula
en América Latina

Fotografía gentileza Johan Fabry.

En este artículo presentamos la experiencia de haber creado un modelo curricular basado en la Computing Curricula [Chang et al., 2001], que luego fue extendido y automatizado para soportar adaptaciones en cualquier país de la región o del mundo, en español o en otros idiomas, respetando las peculiaridades de cada lugar pero siguiendo las recomendaciones internacionales.

Ernesto Cuadros-Vargas
Doctor en Ciencia de la Computación, ICMC, Universidad de Sao Paulo (2004). Fundador y miembro de la Sociedad de Computación Peruana. Secretario Ejecutivo del Centro Latinoamericano de Informática (CLEI) (2009-2010) y miembro del Educational Activities Board de IEEE (2006-2009).

INTRODUCCIÓN

Todos los que hemos tenido la suerte de vivir en esta época podemos observar el avance de la ciencia y tecnología con una mezcla de sentimientos que van desde la admiración, por la cantidad de novedades que aparecen todos los días, pasando por el orgullo de ser protagonistas y llegando a una inevitable sensación de susto por la avalancha acelerada de información que se genera a diario y que parece venirse encima de todos nosotros.

Existen varios factores que han provocado esta aceleración entre los cuales debemos obligatoriamente citar la aparición del computador, la Ciencia de la Computación e Internet. El computador, que fue creado con fines bélicos, pronto fue participando en diversas áreas del conocimiento debido a su poder de resolver problemas que el ser humano demoraba mucho en hacer manualmente. La Ciencia de la Computación, que nació con participación de diversas áreas profesionales, no demoró mucho en ganarse un espacio y hacer sentir que es una disciplina que vino para quedarse y cambiar el mundo tal y como era visto hace seis décadas. Internet, a su vez, se ha convertido en el medio de comunicación de las nuevas generaciones por excelencia. Hoy en día hablamos de los nativos digitales para referirnos a aquellos niños que nacieron

en medio de este mundo digital. Para ellos hablar de tecnología es tan natural como respirar.

Estos tres inventos del ser humano han provocado un círculo virtuoso donde se genera una gran cantidad de información que afecta a todas las áreas del conocimiento, entre las cuales también se encuentra la propia Ciencia de la Computación.

En este artículo presentaremos una propuesta para poder acompañar este avance en el ámbito latinoamericano, siguiendo la filosofía japonesa de imitar, igualar y superar.

SOBRE LA CARRERA EN AMÉRICA LATINA

El área de computación llegó a América Latina como una novedad importada necesaria. La mayoría de las primeras carreras de este tipo en nuestra región se remontan a la década de los '70 y llegan con diversas tendencias y nombres como Ciencia de la Computación (influencia estadounidense), Informática (influencia europea) e Ingeniería de Sistemas, cuyo origen no tiene un claro consenso pero parece referirse también al área de computación.

La Ingeniería de Sistemas es, probablemente, la más confusa por la falta de consenso en definir de forma clara su identidad. Existen mezclas de nombres generadas por diversas influencias. En los EE.UU. se encuentran instituciones como INCOSE (International Council on Systems Engineering, http://www.incose.org) y el ISSS (International Society for the Systems Sciences, http://isss.org), además de las carreras denominadas System Engineering no vinculadas a computación, como la de Massachusetts Institute of Technology (MIT) que presenta su Engineering Systems Division (http://esd.mit.edu). Un reflejo de la falta de consenso en relación a este tema es que la agencia acreditadora ABET (http://www.abet.org) clasifica a la carrera Systems Engineering en el rubro de Others, mientras INCOSE trabaja para que tenga su propio espacio.

Por si fuera poco, existen combinaciones de nombres como Ingeniería de Computación y Sistemas, Ingeniería Informática y Sistemas, etc. los cuales, en muchos casos, tienen el mismo nombre pero se refieren a una carrera distinta. Asimismo hay países donde la carrera se da como licenciatura y en otros como Ingeniería. Una opinión personal del autor es que toda esta confusión es sólo una señal para que continuemos trabajando por aumentar el espacio de la disciplina Computación, frente a otras carreras más antiguas y tradicionales.

Los responsables de ayudar a cambiar esta situación somos, en primer lugar, aquellos involucrados directamente en la formación de profesionales. Necesitamos acortar la distancia en relación a otras regiones y ese es el espíritu de esta propuesta.

SOBRE LA PROPUESTA INTERNACIONAL

A nivel mundial existen carreras que podríamos denominar "tradicionales" como Ingeniería Civil o Medicina, sobre las que dificilmente se tendrían discrepancias acerca de los cursos básicos que hay que dictar para un estudiante universitario. Esto ocurre debido a la antiguedad de esas carreras y a la madurez que ya tienen, así como a una cierta estabilidad y consenso en sus contenidos.

En el caso de la Ciencia de la Computación y todo lo referente al hardware estamos hablando de disciplinas que tienen un poco más de medio siglo de existencia pero han tenido un crecimiento de tipo exponencial. Ambas áreas, hardware y software, se han acelerado mutuamente. Por esta razón, desde hace varias décadas los profesionales de la Ciencia de la Computación, agrupados en instituciones como la ACM y la Sociedad de Computación del IEEE, trabajan arduamente en darles un contenido estándar a las carreras del área.

Los primeros esfuerzos datan de la década de los '60, en las que la ACM publicó el documento [ACM65], que fueron continuados en [COSINE67] y [ACM68]. El crecimiento del área en los siguientes años provocó que esta última recomendación quedase rápidamente obsoleta generando entonces las recomendaciones [EC77, ACM78]. Éstas son actualizadas cada diez años aproximadamente, llegando a [Tuc91] y [CS2001]. La previsión es que deberíamos tener una siguiente actualización en 2011. Pero la velocidad con que las cosas cambian ha obligado a acortar el tiempo para el siguiente reporte y ya vemos actualizaciones en [CC2005] y especialmente en [CS2008].

Basados en todos estos cambios tenemos un gran desafío por delante: ¿Cómo acortar la distancia en nuestra región considerando que muchas veces tenemos menos recursos? La respuesta está en nuestra propia área: utilizar de forma más eficiente la propia Ciencia de la Computación, para poder generar nuevas curricula que respeten las propuestas internacionales, pero que al mismo tiempo nos permitan adaptarlas a las exigencias de cada una de nuestras instituciones.

Los documentos [CS2001, CS2008] presentan los siguientes puntos positivos:

- Presentan un contenido muy detallado del Cuerpo de Conocimiento recomendado para una carrera en Ciencia de la Computación.
- Presentan recomendaciones generales sobre la secuencia de cursos básicos de la carrera.
- El equipo involucrado dispone de los recursos adecuados para generar una recomendación de esta magnitud.
- Si bien es cierto que ambos documentos tienen una fuerte participación estadounidense, también es cierto que no resulta difícil adaptarlos a otras regiones del mundo debido a que la Computación es única y los conceptos dictados son similares.
- El documento [CS2008] hace un especial énfasis en algunos cambios como agregar contenido referente a Seguridad y Paralelismo de forma transversal en toda la carrera y no sólo como cursos aislados. Esto significa que, por ejemplo, conceptos de paralelismo deben ser tocados con diferentes niveles de profundidad, desde que el alumno entra

al primer semestre y presente de forma transversal en cursos como algoritmos, programación, ingeniería de software, base de datos, entre otros.

A pesar de ser un excelente documento de referencia para carreras de pregrado en Ciencia de la Computación, existen varios aspectos que aún no han sido cubiertos por la forma de trabajo que se tuvo. Entre estos puntos podemos mencionar:

- Para la acreditación es necesario relacionar cada curso con un grupo de resultados esperados. Este documento menciona el punto pero no muestra algún tipo de tabla donde sea posible observar esta relación curso por curso.
- La acreditación también solicita que sea posible analizar: ¿Cuántas horas y en qué curso se cubre cada uno de los tópicos del Cuerpo de Conocimiento? sin tener que analizar manualmente cada uno de los cursos de la carrera. Esta información representa una tabla de millares de celdas que contiene las más de cien unidades del Cuerpo de Conocimiento vs todos los cursos de la carrera. Este tipo de información existe pero habría que recurrir al análisis individual y manual de cada curso, a fin de detectar si cumplimos con el mínimo de horas para cada tópico.
- No ofrece una distribución de créditos por áreas que ayude a entender el espectro de una determinada carrera, para poder compararlo con las recomendaciones de máximos y mínimos de créditos por áreas que pueden ser observadas en [CC2005].
- Nuestras universidades suelen ofrecer la malla curricular en forma de cursos clasificados por semestres. Sin embargo, la visión gráfica de los cursos interconectados a través de prerrequisitos ayuda mucho a que el alumno entienda qué líneas forman el camino crítico en la carrera y a cuáles hay que prestarle mayor atención. La consecuencia de esto es que podemos reducir la deserción a través de una mayor cantidad de información de carácter preventivo.
- La propuesta de la [CS2008] tiene un excelente contenido. Pero al mismo tiempo es un documento grande escrito manualmente en LaTeX. Considerando la importancia del documento, y su impacto en nuestra sociedad, lo ideal sería poder generar toda la información antes mencionada de forma automática a partir de la información contenida en el cuerpo de conocimiento y en los cursos.

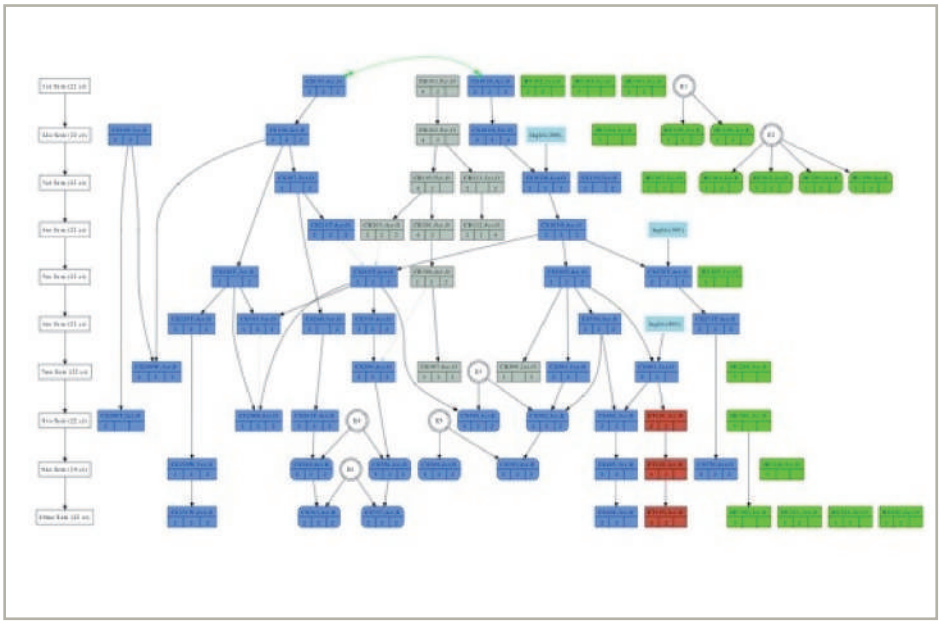


Figura 1: Visualización malla curricular.

SOBRE NUESTRA PROPUESTA

Todos estos cambios desafían frontalmente la forma en que planteamos y enseñamos en una carrera de Ciencia de la Computación. Considerando las ventajas en el documento [CS2008] y sus predecesores, la idea fue crear un programa que pudiese generar, de forma muy rápida, toda una malla curricular con las siguientes características:

- Contenido sólido coherente con el cuerpo de conocimiento de Ciencia de la Computación.
- Actualización respecto a los cambios planteados en [CS2008].
- Visión gráfica de la malla curricular para poder detectar cuellos de botella, caminos críticos, etc., como por ejemplo en la Figura 1. Este grafo es generado utilizando el programa.

Semestre	CS	HU	ET	CB	Total
Primer	8	9		5	22
Segundo	12	5		5	22
Tercero	10	4		8	22
Cuarto	8			14	22
Quinto	15	3		4	22
Sexto	22				22
Séptimo	13	3		6	22
Octavo	16	3	3		22
Noveno	14	3	2		19
Décimo	11	9	2		22
Total	129	39	7	42	

- Distribuciones de cursos por áreas (Figura 2) y por niveles (Figura 3).
- Generar los gráficos comparativos del espectro de una carrera en relación a las propuestas internacionales como Computer Science (CS) (Figura 4), Computer Engineering (CE) (Figura 5), Information Systems (IS) (Figura 6), Software Engineering (SE) (Figura 7), o Information Technology (IT).
- Distribución tabular de cursos por áreas como Ciencia de la Computación (CS), Humanidades (HU), formación de Empresas de Base Tecnológica (ET) y Ciencias Básicas (CB).

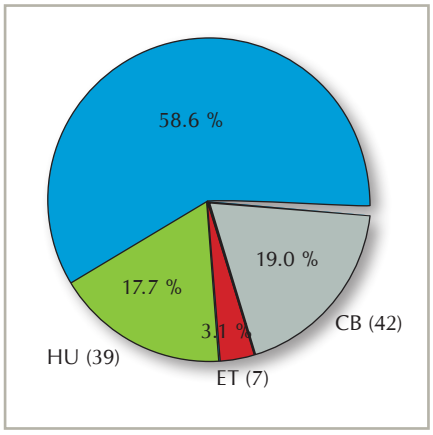


Figura 2: Distribución de créditos por área.

- Distribución de tópicos del Cuerpo de Conocimiento por curso para acreditación y control interno del programa profesional. Esta tabla no puede ser mostrada debido a que supera las nueve mil celdas.
- El docente no entrega el sílabo de cada curso a la jefatura del departamento. Por el contrario, el programa de generación de esta malla también genera los sílabos. Y los cambios en el contenido deben previamente ser coordinados con los docentes del área considerando la visión gráfica de la malla presentada en la Figura 1.
- Para un seguimiento efectivo del avance y desempeño docente, el programa también crea páginas Web para cada curso, de tal forma que el avance es controlado de manera efectiva y los problemas detectados se pueden corregir con rapidez. Esta parte aún no está concluida en 100 por ciento.
- El programa también genera un libro de sílabos, uno con las descripciones cortas de cada curso y un tercero con el listado bibliográfico por curso. Todo esto permite coordinar con la biblioteca una sola vez y para toda la carrera en caso de no existir un libro disponible.

En base a esta propuesta es posible acortar mucho el tiempo dedicado a este tipo de tareas de gestión de la carrera y aumentar la calidad y el control de lo que se dicta, así como detectar problemas antes de que sucedan.

EXTENDIENDO EL MODELO PARA CUALQUIER OTRA CARRERA

Gracias a la abstracción inherente a nuestra formación en Ciencia de la Computación es fácil entender que el programa sería facilmente adaptable a diversas otras carreras. En este sentido, para extender este modelo necesitamos seguir los siguientes pasos:

- Definir en forma de macros LaTeX el Cuerpo de Conocimiento del área.
- Lista de resultados Outcomes de acuerdo a la acreditación del área.
- Definir el contenido de cada curso en función del cuerpo del conocimiento respectivo. Es aquí donde relacionaremos cada curso con un conjunto de resultados esperados.
- Redactar la información básica de la carrera tal como: visión, misión, títulos otorgados, relevancia en la sociedad, etc.

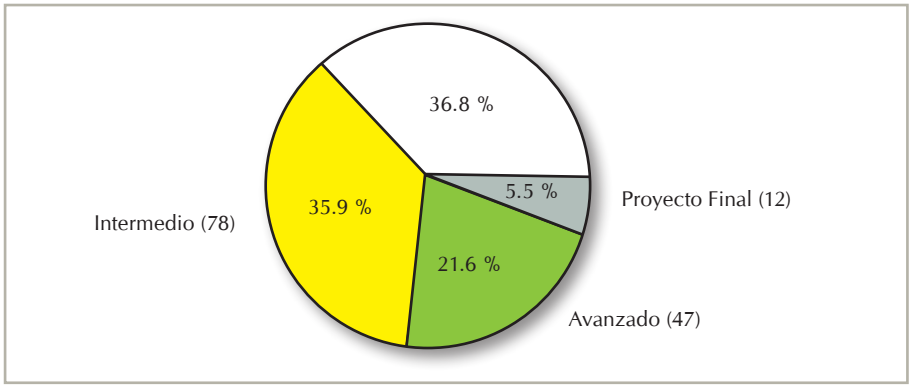


Figura 3: Distribución de créditos por nivel.

Probablemente el mayor problema sería pedir que otras carreras llenasen el contenido de los cursos en LaTeX. Para resolver ese problema se está trabajando en una interfase vía Web que facilite la interacción de colegas de otras carreras.

CONCLUSIONES Y TRABAJOS FUTUROS

Este documento presenta una propuesta para generar y mantener actualizada una malla curricular de acuerdo a cualquier propuesta internacional.

La cantidad de tiempo invertido en llenar la información para una nueva carrera reduce al orden de algunos pocos días el trabajo que antes podía tomar fácilmente meses.

El control que es posible tener sobre la carrera, basado en los diversos indicadores, abre una gran cantidad de posibilidades y nos ayuda a detectar tempranamente múltiples problemas y mejorar la calidad de la educación ofrecida.

Una característica que se adicionará es que, en base al seguimiento por curso, será posible recolectar los datos con el objetivo de generar estadísticas históricas y analizar el desempeño académico de un docente en todos los cursos que dicta.

Asimismo será posible comparar todos los cursos que dicta un único profesor (a), a fin de analizar en cuáles se desempeña mejor y detectar posibles puntos débiles para reforzar.

Teniendo tabulados los resultados esperados por cada curso es posible comparar de manera efectiva si el (la) docente está cumpliendo con los mismos, y generar indicadores que retroalimenten a los propios docentes y permitan que él/ella mejore su cátedra.

La visión gráfica de la malla permite detectar posibles incoherencias en la malla curricular, lo que es mucho más dificil de hacer sobre una visión tabular tradicional de cursos distribuidos por semestre. BITS

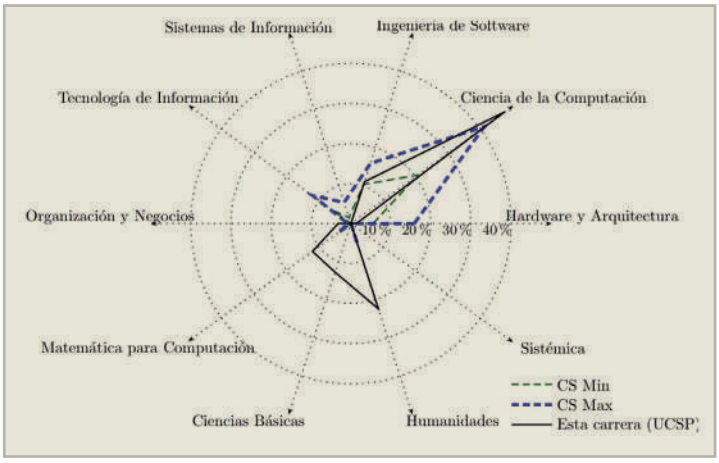


Figura 4: Comparación de la malla generada contra la recomendación de CS.

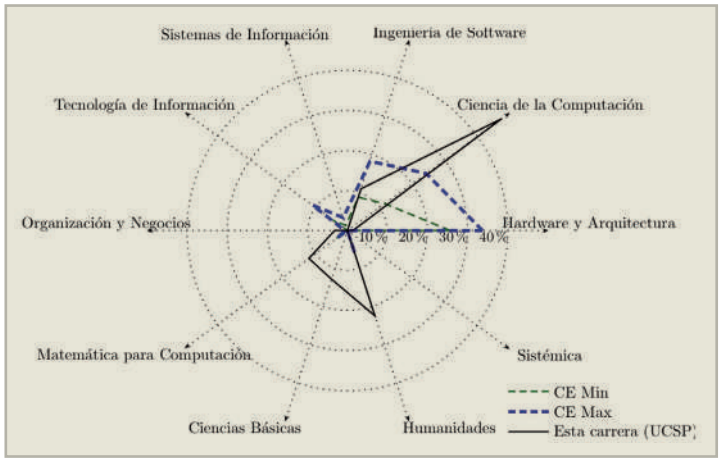


Figura 5: Comparación de la malla generada contra la recomendación de CE.

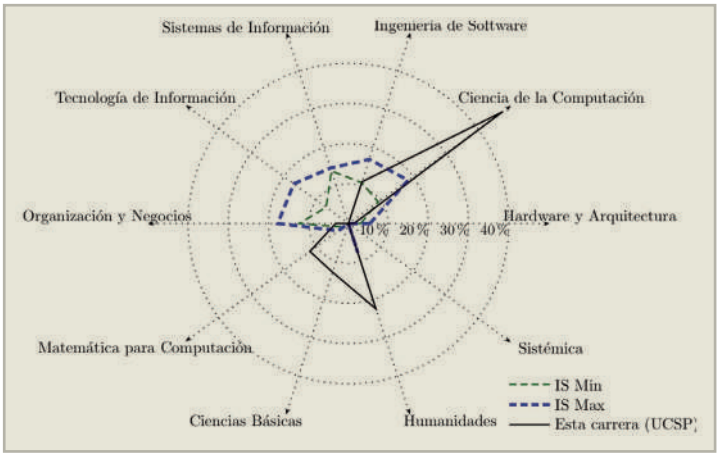


Figura 6: Comparación de la malla generada contra la recomendación de IS.

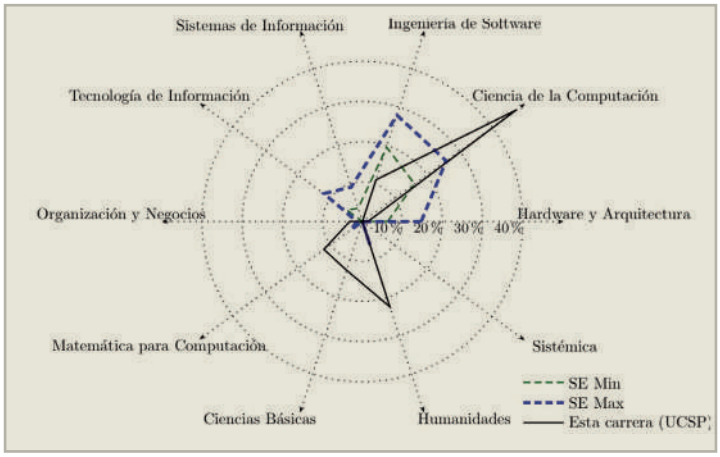


Figura 7: Comparación de la malla generada contra la recomendación de SE.

REFERENCIAS

[ACM65] ACM (1965). Curriculum Committee on Computer Science. An Undegraduate Program in Computer Science. Communications of the ACM, 8(9):543–552.

[ACM68] ACM (1968). Curriculum Committee on Computer Science. Curriculum '68: Recommendations for the Undegraduate Program in Computer Science. Communications of the ACM, 11(3):151–197.

[ACM78] ACM (1978). Curriculum Committee on Computer Science. Curriculum '78: Recommendations for the Undegraduate Program in Computer Science. Communications of the ACM, 22(3):147–166.

[CS2008] Cassel, L., Clements, A., Davies, G., Guzdial, M., McCauley, R., McGettrick, A., Roberts, E., Sloan, B., Snyder, L., Tymann, P., and Weide, B. W. (2008). Computer science curriculum 2008: An interim revision of cs2001. Technical report, ACM/IEEE-CS.

[CS2001] Chang, C., Denning, P. J., II, J. H. C., Engel, G., Sloan, R., Carver, D., Eckhouse, R., King, W., Lau, F., Mengel, S., Srimani, P., Roberts, E., Shackelford, R., Austing, R., Cover, C. F., Davies, G., McGettrick, A., Schneider, G. M., and Wolz, U. (2001). Computing Curricula 2001 Computer Science. Technical report, ACM/IEEE, http://www.computer.org/education/cc2001/steelman/cc2001/index.htm. Last visited March 2004.

[COSINE67] COSINE Committee (1967). Computer Science in Electrical Engineering. Commission on Engineering Education, Washington, DC.

[EC77] IEEE-CS (1977). A Curriculum in Computer Science and Engineering. Technical Report, IEEE-CS. Education Committee of the IEEE-CS. Publication EHO119-8.

[CC2005] Shakelford, R., Cross, J. H., Davies, G., Impagliazzo, J., Kamali, R., LeBlanc, R., Lunt, B., McGettrick, A., Sloan, R., and Topi, H. (2005). Computing curricula 2005. Technical report, ACM/IEEE, http://www.acm.org/education.

[Tuc1991] Tucker, A. B., Barnes, B. H., Aiken, R. M., Barker, K., Bruce, K. B., Cain, J. T., Conry, S. E., Engel, G. L., Epstein, R. G., Lidtke, D. K., Mulder, M. C., Rogers, J. B., Spafford, E. H., and Turner, A. J. (1991). Computing curricula '91. Technical report, Association for Computing Machinery and the Computer Society of the Institute of Electrical and Electronics Engineers.

Universidad de Chile

Nuevo Currículo de Ingeniería

A partir de 2007, los alumnos que ingresan al Plan Común de Ingeniería en la Facultad de Ciencias Físicas y Matemáticas se encuentran con un nuevo Plan de Estudios y un número de innovaciones en los métodos de enseñanza y aprendizaje. Este artículo describe las razones del cambio y los lineamientos principales que lo guían y que inciden en el diseño de la carrera de Ingeniería Civil en Computación.

Patricio Poblete
Director de la Escuela de Ingeniería y Ciencias de la Facultad de Ciencias Físicas y Matemáticas, Universidad de Chile. Profesor Titular, DCC, de la misma Universidad. Ph.D. (1982), M.Sc. (1977), Computer Science, University of Waterloo.
ppoblete@dcc.uchile.cl

INTRODUCCIÓN

La Escuela de Ingeniería y Ciencias de la Universidad de Chile es la más antigua de su tipo en el país y es considerada una de las escuelas líderes. Los alumnos que ingresan a ella cada año se cuentan entre los mejores que produce la Enseñanza Media. Y los profesionales que egresan suelen desempeñarse exitosamente y ocupar posiciones destacadas en la industria.

Lo anterior parecería indicar que no hay motivos para plantear reformas profundas en la docencia de la Escuela, pero existen razones importantes que hacían necesario emprender esta tarea.

En primer lugar, toda escuela que aspire a ser líder o mantenerse como tal, debe necesariamente analizar su quehacer en busca de oportunidades para mejorar, dado el ambiente competitivo en que se desenvuelve, tanto a nivel nacional como también –crecientemente– internacional.

En segundo lugar, diversos análisis realizados al interior de la Facultad sugerían que había oportunidades importantes para mejorar la docencia y abordar de esta manera debilidades que se percibían en sus egresados. En particular, algunos de los temas en que había consenso que se necesitaba mejorar eran:

- Lograr un aprendizaje más profundo y duradero, robusteciendo la capacidad de aplicar los conocimientos adquiridos.

- Generar oportunidades tempranas de contacto con los problemas y métodos de la Ingeniería en forma práctica a través de proyectos, fomentando el desarrollo de la creatividad, la invención e innovación.
- Robustecer las habilidades personales e interpersonales de los estudiantes: comunicación, capacidad de trabajo en equipo, liderazgo.

A fines de 2002, la Facultad aprobó el inicio de un trabajo de reforma docente que comenzó en 2003 a través de una Comisión de Desarrollo Docente designada para este efecto, con represententes de todos los departamentos, los alumnos y las organizaciones ligadas a la profesión. Esta Comisión trabajó durante varios años generando propuestas de reforma para el Plan Común, las licenciaturas y finalmente las especialidades. Y el nuevo Plan de Estudios así elaborado comenzó a aplicarse a partir de 2007, con los alumnos ingresados en dicho año.

El tiempo que tardó la elaboración y aprobación de esta propuesta de reforma sin duda excedió lo que se había planteado originalmente. Pero la Facultad entendió que una reforma como ésta debía ser bien analizada y ampliamente consensuada para que fuera real y no sólo formal. La tarea planteada resultó ser mucho más difícil que lo que se pensó en un principio. Sin embargo bastante más llena de oportunidades. En particular, desde ya muy al comienzo, la Comisión se dio cuenta que no bastaba sólo con cambiar los planes de estudios, sino que se debía llegar a impactar lo que ocurría en la sala de clases, entre el profesor y sus alumnos. Es decir, el tema de las metodologías docentes debía ser parte integrante de la propuesta de cambio.

EL CONTEXTO INTERNACIONAL

Al buscar si existían experiencias similares en el mundo, la Comisión se encontró rápidamente con que el tema de la reforma de la enseñanza de la Ingeniería era una materia que estaba siendo abordada en muchos lugares.

En Europa, la Declaración de Bolonia había puesto en marcha un amplio proceso de revisión curricular en todo ese continente. Orientada en un comienzo a fomentar la movilidad de estudiantes y profesionales, dicha Declaración planteó redefinir la estructura de los estudios superiores de modo que, de manera universal, se empleara el esquema Bachelor-Master-Doctorado vigente en el mundo anglosajón y en numerosos países de Asia. Esto de inmediato forzó a muchos países a redefinir completamente sus planes de estudios y, junto con ellos, los programas de las asignaturas. Asimismo, la necesidad de poder certificar las habilidades adquiridas, algo necesario para la movilidad, hizo que en estos procesos de renovación curricular se adoptaran fuertemente los enfoques basados en competencias.

La experiencia europea es sin duda un referente fundamental a la hora de pensar en reformar una carrera universitaria. Y en nuestro caso ha tenido un impacto importante, especialmente en lo que se refiere a la estructura de grados académicos que plantea. Pero por otra parte, al tratarse de una obra en curso, no hay aún resultados observables que ayuden a evaluar los resultados finales. Esta es una de las razones por las cuales en nuestro análisis tuvieron mucho mayor impacto las experiencias de reforma norteamericanas.

En Estados Unidos, el cambio en los criterios de acreditación introducidos en ABET 2000[1], y sus criterios "a-k", hizo que muchas escuelas debieran revisar sus planes de estudios y metodologías, y esto fue apoyado por la National Science Foundation al financiar un número de "coaliciones" para el mejoramiento de la enseñanza de la Ingeniería. De particular interés resultó la experiencia de la "Foundation Coalition"[2]. Esto mismo motivó también a la Franklin W. Olin Foundation a establecer el Franklin W. Olin College of Engineering[3], con la misión explícita de ser innovador en la enseñanza y aprendizaje de la Ingeniería. Por su parte, escuelas tradicionales como MIT también estaban llevando a cabo procesos de innovación curricular importantes. De estos últimos tres fueron particularmente relevantes: primero, el trabajo pionero del profesor Woodie Flowers[4] en el desarrollo de la creatividad y capacidad de innovación de los estudiantes. Segundo, el cambio en la enseñanza de la Física materializado en las salas TEAL[5] (Technology Enabled Active Learning). Y tercero, el enfoque de rediseño curricular llamado CDIO[6], originado en el Departamento de Ingeniería Aeronáutica y Aeroespacial de MIT y luego adoptado por un gran número de escuelas en todo el mundo.

Finalmente, en el ámbito de las metodologías de enseñanza y aprendizaje, resultó muy influyente para nosotros el trabajo del profesor Eric Mazur de Harvard, con su metodología de "peer instruction"[7] para el aprendizaje colaborativo en grandes salas de clases. Y del grupo ALE[8] (Active Learning in Engineering Education), que a través de conferencias anuales fomenta el uso de todos los métodos que involucren activamente a los estudiantes en su aprendizaje.

EL PERFIL DEL EGRESADO

En la definición del perfil del egresado se buscó robustecer las fortalezas tradicionales de la Escuela, que incluyen una formación muy sólida en los fundamentos de la ciencia y las matemáticas, junto con abordar las áreas en que se había detectado que los egresados presentaban debilidades. El resultado fue un perfil en que se plantea el siguiente conjunto de logros para los estudiantes de la Facultad:

- Deben adquirir un fuerte dominio de las matemáticas y las ciencias básicas, y ser capaces de aplicar estos conocimientos en los cursos donde estos se requieren.
- Deben adquirir una fuerte formación en ciencias de la Ingeniería y tener dominio de la tecnología actual.
- Deben adquirir capacidad para diseñar experimentos, obtener, utilizar e interpretar datos.
- Deben tener la capacidad de plantear y resolver problemas de Ingeniería, especialmente enfrentar problemas abiertos o que requieran un enfoque multidisciplinario.
- Deben desarrollar la capacidad de diseño en Ingeniería.
- A lo largo de todo el Plan de Estudios, deben cultivar y ejercitar la capacidad de autoaprendizaje y tomar conciencia de la importancia de mantener este hábito una vez egresados.
- Deben adquirir desde temprano la habilidad de trabajar en equipo, incluyendo aquellos multidisciplinarios.

- Deben adquirir la capacidad de comunicarse en forma efectiva, tanto oral como escrita, en castellano como en inglés. Es importante también que tengan capacidad de expresar sus ideas en forma gráfica. Debe haber cursos orientados específicamente a esta adquisición de competencias. Pero es fundamental que ellas se ejerciten a lo largo de todo el Plan de Estudios.
- Deben adquirir competencia en análisis económico y administración, independientemente de la especialidad que sigan.
- Deben reconocer la importancia de un comportamiento ético tanto en los estudios como en la posterior vida profesional, y actuar en consecuencia.

LA METODOLOGÍA DOCENTE

Junto con el nuevo Plan de Estudios, la Escuela está realizando un esfuerzo importante por apoyar el uso de metodologías activas en su docencia.

Un ejemplo importante de esto es la nueva Sala Galileo para la enseñanza de la Física en primer año. Esta sala, inspirada en las salas TEAL de MIT, rompe la distinción tradicional entre cátedra y laboratorio y provee un ambiente adecuado para el aprendizaje colaborativo. En esta sala los alumnos, sentados en mesas circulares con nueve alumnos cada una, pueden recibir parte de una cátedra, y luego experimentar lo aprendido con "kits" disponibles en cada mesa, analizar los resultados usando herramientas como MATLAB, utilizar softwar de simulación y visualización, etc. Esto ha ido acompañado de una completa revisión de la pedagogía usada en este curso, y la experiencia obtenida en las primeras versiones de la cátedra se ha utilizado para mejorar las versiones posteriores.

Otro ejemplo de aprendizaje activo son las nuevas clases auxiliares de los cursos de matemáticas de primer semestre. En estas clases los estudiantes dejan de ser espectadores de un profesor auxiliar resolviendo problemas en la pizarra, y en lugar de eso deben involucrarse activamente en la resolución guiada de problemas, trabajando en equipos.

Estas, así como muchas otras innovaciones metodológicas que están poniendo en práctica nuestros profesores, requieren para tener éxito un apoyo de personal especializado en pedagogía, para lo cual se ha establecido un Área de Desarrollo Docente. Este grupo está encabezado por un profesor de Ingeniería, y cuenta con un staff experto en pedagogía (a nivel de Magíster en Educación). Su trabajo aborda desde el apoyo personalizado para académicos que lo requieran, hasta la organización de actividades grupales de desarrollo docente, tanto para profesores como para profesores auxiliares. Entre las actividades se destacan los talleres para académicos que se inician en la docencia, de aprendizaje basado en problemas y de manejo de la voz en la sala de clases.

En forma transversal a todo lo anterior, la Escuela ha adoptado masivamente la plataforma "U-Cursos", desarrollada en la misma Facultad con importantes aportes de los mismos estudiantes, como la base tecnológica para su docencia. Gracias a "U-Cursos", todas las cátedras disponen automáticamente de un sitio Web a través del cual el profesor y los alumnos pueden publicar material docente, comunicarse a través de mail y foros, entregar tareas, publicar notas y todas las demás actividades asociadas al funcionamiento de cada cátedra.

LA DURACIÓN DE LOS ESTUDIOS

La duración, tanto nominal como efectiva de los estudios ha sido materia de bastante debate y preocupación. Las innovaciones docentes puestas en práctica por la Escuela van, entre otras cosas, orientadas a mejorar los indicadores de éxito académico de los estudiantes y acercar así la duración real a la nominal. Respecto de ésta última, la Facultad considera que la profundidad y amplitud que requiere la formación que entrega, incluyendo la exigencia de una memoria de título, requieren un número de años gruesamente similar a los seis tradicionales en Chile. Reconociendo, sin embargo, que esto puede variar de una especialidad a otra, se estableció un rango entre once y doce semestres, teniendo cada departamento libertad para fijar la duración de su carrera dentro de estos márgenes.

Lo anterior permitió, por ejemplo, que el Departamento de Ciencias de la Computación diseñara su carrera de Ingeniería Civil en Computación con una duración total de once semestres, mientras otros departamentos se acercaron más a los doce semestres tradicionales.

Un importante tema pendiente, que debería ser abordado en el futuro cercano, es el de la relación entre el título de ingeniero y el grado de magíster (o máster). Después de Bolonia, el mundo de manera casi universal se ha orientado a utilizar los grados académicos de Bachelor, Máster y Doctorado como las medias de avance en la formación de una persona. En el caso europeo, se espera que el máster se alcance al cabo de cinco años (ya sea en un esquema "3+2" o "4+1"), así que es natural plantear la pregunta de si un ingeniero de nuestra Facultad, con una formación de cinco años y medio, o de seis, puede o no considerarse como equivalente a un máster.

CONCLUSIONES

Después de un largo camino, nuestra Facultad avanza en la implantación de un nuevo Plan de Estudios asociado a una renovada metodología docente. Una de las conclusiones de este proceso es que este tipo de análisis no puede hacerse separado por grandes períodos de tiempo, sino debe constituirse en un proceso de mejoramiento continuo. En el par de años de vigencia del nuevo Plan ya podemos apreciar mejoras importantes. Pero si tenemos éxito en la consolidación de esta cultura de mejoramiento continuo, sin duda veremos en el futuro avances todavía más significativos. BITS

[1] ABET, Criteria for Accrediting Engineering Programs, http://www.abet.org/Linked%20Documents-UPDATE/Criteria%20and%20PP/05-06-EAC%20Criteria.pdf
[2] http://clte.asu.edu/active/main.htm
[3] http://www.olin.edu
[4] http://meche.mit.edu/people/faculty/index.html?id=26
[5] http://icampus.mit.edu/teal/
[6] http://www.cdio.org
[7] http://mazur-www.harvard.edu/research/detailspage.php?ed=1&rowid=8
[8] http://ale2009.ac.upc.edu/

EDUCACIÓN EN INFORMÁTICA

En FCFM de la U. de Chile
Nuevo Plan de Estudios de Computación

Nancy Hitschfeld
Profesora Asociada, DCC, Universidad de Chile. Dr. in Applied Sciences, ETH-Zurich (1993), Magíster en Computación, DCC, Universidad de Chile (1987). Líneas de investigación: Computación Gráfica, Tecnologías en Mallas Geométricas y Aplicaciones, Diseño Geométrico Asistido por Computador.
nancy@dcc.uchile.cl

José Miguel Piquer
Profesor Asociado, DCC, Universidad de Chile. Doctor en Computación, École Polytechnique, París. Director Técnico de NIC Chile y Director de NIC Labs.
jpiquer@dcc.uchile.cl

Juan Alvarez Rubio
Académico DCC, Universidad de Chile. Master of Mathematics (Computer Science), University of Waterloo. Ingeniero de Ejecución en Procesamiento de la Información, Universidad de Chile.
jalvarez@dcc.uchile.cl

Este artículo presenta el nuevo Plan de Estudios, tanto de la carrera de Ingeniería Civil en Computación (ICC) como de una especialización secundaria en Computación para los alumnos de otras especialidades. La actualización surge en el contexto de la definición de un nuevo perfil para el ingeniero civil formado en la Facultad de Ciencias Físicas y Matemáticas (FCFM) de la Universidad de Chile, con el apoyo del *Proyecto MECESUP UCH0403*.

Después de más de quince años de experiencia con el plan de estudios anterior era imperativo actualizar las carreras, y en particular la enseñanza en Computación, considerando la masificación de las tecnologías y la diversidad de áreas donde se aplican. Asimismo, en el marco de los recientes procesos de autoevaluación y acreditación de la carrera, los empleadores aportaron valiosa información respecto de las fortalezas y debilidades de los egresados.

Los fundamentos principales de los cambios propuestos son mayor flexibilidad (los alumnos escogen áreas) y mayor amplitud (incluye habilidades de otros ámbitos). El propósito de manejar y mejorar las relaciones humanas se logra desarrollando destrezas transversales (expresión oral y escrita, trabajo en grupo, etc.).

En consecuencia, el nuevo Plan de Estudios de la ICC presenta dos elementos novedosos:

- Obtener la licenciatura en Computación, en ocho semestres, con una segunda especialización (*minor*) elegida libremente por el alumno dentro de un conjunto de áreas de la ciencia e ingeniería tales como: astronomía, geología, energías renovables, biotecnología, minería, etc.
- Obtener la ICC (en once semestres) con una especialización elegida por el estudiante entre ingeniería de software, tecnologías

de la información y comunicaciones, ciencia e ingeniería computacional, etc.

Cabe señalar que la formación que se entrega durante los dos últimos años es equivalente a un grado de Magíster. En consecuencia, si el alumno realiza una tesis que dura dos semestres en vez del proyecto o memoria de título que dura uno, obtiene el grado de Magíster en Ciencias de la Computación junto al título profesional de Ingeniero Civil en Computación.

Los alumnos de las otras especialidades adquieren una formación básica en Computación que les permite enfrentar problemas, diseñar soluciones y resolverlos computacionalmente. Además, para complementar su formación se creó un *minor* en Computación que les brinda la posibilidad de adquirir habilidades avanzadas para resolver, de manera más efectiva y eficiente, los problemas que se presentan en su propia especialidad.

A continuación se describen las consideraciones más importantes que se tuvieron en cuenta al diseñar el nuevo perfil de nuestros egresados, y se señala en qué consiste el actual Plan de Estudios de la carrera ICC y el *minor* en Computación.

I.- LA CARRERA DE INGENIERIA CIVIL EN COMPUTACIÓN

¿Qué perfil queremos para nuestros egresados?

Hemos definido un perfil que exige que el alumno adquiera tanto habilidades técnicas específicas de su especialidad como destrezas transversales comunes a todas las especialidades. Este nuevo perfil sigue los estándares de la iniciativa CDIO (http://www.cdio.org), que define un marco acerca de las habilidades fundamentales para los ingenieros de la próxima generación.

Se consideran como habilidades para resolver problemas en el área de las ciencias básicas, comunes a toda ingeniería, aquellas útiles para solucionar problemas específicos de la profesión. Además de las habilidades que permiten gestionar proyectos y trabajar de forma individual y en grupo. Estos dos primeros tipos de habilidades formaban ya parte en gran medida del plan anterior. Y dado que fueron evaluadas muy positivamente por nuestros egresados y sus empleadores, se mantuvieron y mejoraron en el nuevo Plan de Estudios. El tercer grupo incluye una serie de habilidades transversales que no estaban explícitas en el plan antiguo, pero que sí fueron mencionadas por nuestros egresados y empleadores.

> Después de más de quince años de experiencia con el plan de estudios anterior era imperativo actualizar las carreras, y en particular la enseñanza en Computación, considerando la masificación de las tecnologías y la diversidad de áreas donde se aplican.

A continuación se resumen las principales habilidades que contempla el perfil (ver lista completa en http://www.dcc.uchile.cl):

I.- Conocimiento sólido en ciencias básicas

- Fuerte dominio de las matemáticas, ciencias básicas y ciencias de la ingeniería, y una comprensión global de los fundamentos de la Ciencia de la Computación, incluidos conocimientos avanzados en una o más áreas y la destreza para aplicarlos de manera rigurosa e integrada.
- Capacidad de diseñar experimentos, obtener, utilizar e interpretar datos.

II.- Conocimiento sólido en tecnologías

- Dominio de técnicas y herramientas modernas necesarias para ejercer la profesión.
- Habilidad para concebir, diseñar, implementar, operar, evaluar y controlar software, sistemas, componentes o procesos, en forma eficiente y creativa, que cumplan con las especificaciones demandadas por el contexto y según restricciones económicas, ambientales, sociales, políticas, éticas, de salud y seguridad, de manufactura y sustentabilidad.
- Capacidad de identificar, formular y resolver problemas computacionales de manera robusta y eficiente, en especial para enfrentar problemas abiertos o que requieren un enfoque multidisciplinario.

III.- Habilidades de gestión personal y grupal

- Capacidad de autoaprendizaje y conciencia sobre la importancia de continuar aprendiendo luego de egresar, manteniéndose permanentemente al día en la ciencia y tecnología de la información.
- Destreza para trabajar en equipo, incluidos aquellos multidisciplinarios, y de desenvolverse adecuadamente en diferentes entornos demostrando atributos de liderazgo. Comunicarse de manera efectiva en diversos contextos, tanto en forma oral como escrita, en castellano e inglés.
- Aptitud de emprender e innovar, con competencia en los fundamentos del análisis económico y de gestión, capaz de agenciar y administrar proyectos informáticos.
- Capacidad de reconocer la importancia de un comportamiento ético en la vida profesional, demostrando honestidad, integridad, responsabilidad hacia la sociedad y el medio ambiente. Y capacidad de reconocer sus propias potencialidades y limitaciones. Manejo de conocimiento sobre temas contemporáneos y una actitud de valoración y respeto hacia otras culturas.

¿Cómo lograr formar profesionales con este perfil?

Al diseñar el Plan de Estudios se definió un plan común de dos años para todas las carreras de la Facultad. Para los dos años siguientes se propuso que el alumno adquiera habilidades que, si así lo desea, le permitan salir al mercado laboral con el grado de Licenciatura en Computación.

El núcleo computacional de la Licenciatura permite obtener destrezas, usando buenas prácticas, para desarrollar software de mediana complejidad, estimar la complejidad de los algoritmos, diseñar experimentos para evaluar desempeño, identificar problemas difíciles así como detectar aquellos que no tienen solución computacional con los modelos de computadores actuales. En esta etapa se enfatiza más el trabajo individual que en grupo. En el último año y medio, y como parte del proceso de obtención del título de Ingeniero Civil en Computación, el estudiante desarrolla las habilidades necesarias para gestionar proyectos, innovar y trabajar en grupo. El alumno tiene la posibilidad de seguir una línea de especialización.

La Licenciatura en Ciencias de la Ingeniería mención Computación

Nuestra Licenciatura se diferencia del resto de las licenciaturas de la Facultad en que es perfectamente "habilitante". Es decir, al terminar este ciclo el estudiante se encuentra preparado para ingresar al mundo laboral en niveles medios de responsabilidad e integrar grupos de desarrollo de software. Siendo más precisos, el Plan de Estudios de la Licenciatura se caracteriza por:

- Poseer un fuerte componente de formación básica (principalmente en matemáticas y física y en menor grado química y economía); más dos cursos de Introducción a la Ingeniería y Taller de Proyecto, en donde los alumnos adquieren habilidades para concebir y desarrollar proyectos de ingeniería junto a las de trabajo en grupo; expresión oral y escrita y cursos de formación integral, los que incluyen el dominio de una segunda lengua (inglés), más ramos del área de humanidades y deportivos. Estos cursos están agrupados en un plan común de cuatro semestres (dos años) para todas las ingenierías. Durante el tercer año los estudiantes extienden su formación básica incluyendo tres cursos de una lista de complementos de formación básica que, en el caso particular de nuestra carrera, incluye temas como computación gráfica, visualización y modelamiento para ingenieros, probabilidades, estadísticas y matemáticas discretas.
- El tercer y cuarto año se componen de once cursos obligatorios y cinco electivos. Los obligatorios están orientados a que el alumno adquiera conocimientos amplios y profundos y habilidades consideradas como parte del núcleo de la disciplina. Los temas cubiertos son: algoritmos y estructuras de datos, computación teórica, análisis y diseño de algoritmos, ingeniería de software, arquitectura de computadores y bases de datos, entre otros. Los problemas a resolver, en general, están orientados a evaluar las habilidades individuales de los alumnos. Los ramos electivos son de elección del alumno y por tanto puede tomarlos de un conjunto coherente de cursos ofrecidos por algún departamento de la Facultad, o en forma selectiva dependiendo de sus gustos personales. Si elige cuatro cursos electivos de un conjunto coherente obtiene un *minor*, que le permite adquirir una subespecialidad. Existe una oferta muy variada de *minors* ofrecida por los otros departamentos de la Facultad (http://escuela.ing.uchile.cl/docencia/*minor*). Junto a lo anterior, el alumno tiene espacio para complementar su formación con cursos de formación integral.

La Especialidad

La etapa poslicenciatura se puede dividir en tres:

- Formación común, orientada a que todos los estudiantes adquieran habilidades más profundas y avanzadas en el área de ingeniería de software y gestión de proyectos. Todos los alumnos, siendo integrantes de un grupo de trabajo, deben ser capaces de enfrentar un problema real, encontrar una solución e implementarla.
- Formación especializada, con cursos electivos orientados a que los estudiantes obtengan una línea de especialización. Algunas de estas líneas son dictadas sólo por profesores del Departamento de Ciencias de la Computación, y entre ellas están: ingeniería de software, computación teórica, manejo de información, interfaz humano-computador, y ciencia e ingeniería computacional. Otras líneas como tecnologías de la información y comunicaciones e inteligencia computacional y robótica, son dictadas junto con profesores de otros departamentos.
- Desarrollo de un proyecto de título individual durante seis meses. Sobre este trabajo el estudiante debe escribir una memoria y presentar su defensa ante una comisión. Si al mismo tiempo desea obtener el grado de Magíster en Ciencias mención Computación, en vez de la memoria debe hacer una tesis de un año y dar la respectiva defensa de tesis también ante una comisión.

II.- COMPUTACIÓN EN LA FACULTAD

Plan Común

El primer semestre del Plan Común contempla la asignatura de Computación, que tiene como propósito desarrollar en los alumnos el razonamiento algorítmico-lógico como una de las dimensiones fundamentales de la capacidad universal para resolver problemas. El objetivo general del curso es que, al final de éste, los alumnos puedan resolver problemas con la ayuda de la Computación (como disciplina). Mientras el objetivo específico es resolver problemas del ámbito de las ciencias físicas y matemáticas programando los algoritmos de solución, tanto en un lenguaje de propósito general (Java) como en uno especializado (Matlab).

Los contenidos contemplan cinco capítulos o unidades: Fundamentos de Programación, Introducción a la Programación Orientada a

Objetos, Manejo de Listas y Tablas de Valores (arreglos), Introducción a la Computación Numérica, Búsqueda y Ordenamiento de Información. Siguiendo los lineamientos generales del nuevo Plan de Estudios, el curso se orienta al aprendizaje de los alumnos, es decir, todas las actividades docentes están centradas en los estudiantes de manera que logren los objetivos del curso.

Complemento a la formación básica

Terminado el plan común, los alumnos pueden inscribir dos cursos adicionales de Computación para obtener una visión más amplia de las áreas que pueden ser relevantes al momento de tomar decisiones en el ejercicio de su profesión. De hecho, uno de los cursos aborda temas fundamentales como modelamiento orientado a objetos, bases de datos, computación paralela y manejo de grandes volúmenes de datos. El otro curso abarca los aspectos más importantes de la computación gráfica, técnicas de visualización y modelamiento aplicado para problemas en ciencias e ingeniería. En ambos cursos los alumnos deben enfrentar problemas, plantear la solución e implementarla en el lenguaje apropiado. Estos dos cursos forman parte también del *minor* en computación del área de aplicaciones a la ingeniería.

Minor en Computación

Así como nuestros alumnos tienen la posibilidad de obtener un *minor* en diversas áreas de la ingeniería y ciencias, para los estudiantes de otras especialidades hemos diseñado un *minor* en Computación. La idea es conocer y entender con mayor profundidad cómo sacar provecho a la Computación en el contexto de su profesión. El alumno aprenderá los conceptos básicos para desarrollar software eficiente de mediana complejidad, entendiendo los fundamentos de la Ciencia de la Computación y utilizando buenas prácticas en el área de su especialidad. Al mismo tiempo, si los problemas son computacionalmente complejos, podrá comunicarse de manera fluida con profesionales especializados en Computación. Para obtener este *minor*, el alumno debe cursar cuatro asignaturas

de Computación, siendo obligatorio un curso de algoritmos y estructuras de datos fundamentales. Los tres cursos restantes pueden ser elegidos de algunas de las siguientes áreas:

- Algoritmos, que incluye cursos de matemáticas discretas, teoría de la computación, diseño y análisis de algoritmos, entre otros.
- Software de sistemas: programación de sistemas, arquitectura de computadores y sistemas operativos, entre otros.
- Ingeniería de software: metodologías de diseño y programación, ingeniería de software para desarrollar software de manera individual y en grupo, entre otros.
- Aplicaciones a la ingeniería: computación gráfica, visualización y modelamiento, programación cercana a la máquina, manejo de grandes volúmenes de datos y geometría computacional, entre otros.

III.- CONCLUSIONES

El nuevo Plan de Estudios mantiene las fortalezas de nuestros actuales ingenieros civiles en Computación: (a) El conocimiento y dominio amplio y profundo de las áreas fundamentales de la Computación así como de las tecnologías de punta del momento. (b) La capacidad de enfrentar nuevos problemas y proponer soluciones adecuadas.

Asimismo, el nuevo Plan permite que el alumno tenga una segunda especialidad en la licenciatura como en el período pos-licenciatura. La idea es formar profesionales que puedan adaptarse fácilmente a desarrollar aplicaciones en distintas áreas de la ciencia, ingeniería y humanidades. La Computación permea todas las áreas del conocimiento y de la vida. Por tanto queremos que nuestros profesionales tengan más oportunidades y egresen mejor preparados para enfrentar los desafíos innovadores de la Computación, en cualquier lugar y tema.

Algunas líneas de especialización ofrecidas poslicenciatura están basadas en las recomendaciones del último currículum de la ACM. En él se reconoce explícitamente el hecho de que en la actualidad una persona no puede tener conocimientos profundos de todas las áreas de la Computación. Nuestro Plan de Estudios (de once semestres) se diferencia del currículum de la ACM (de ocho semestres) en que los alumnos realizan una licenciatura común, con mucho mayor énfasis en matemáticas y física, y en que el núcleo común de los cursos de Computación es mucho más profundo y amplio.BITS

EDUCACIÓN EN INFORMÁTICA

Programación Robots Lego: Aprender Jugando en el DCC

Jérémy Barbay
Profesor Asistente, DCC, Universidad de Chile. Master y PhD en Computer Science, Orsay, France (2002); Bachelor of Science in Mathematics, Rouen, Francia (1997).
jbarbay@dcc.uchile.cl

Johan Fabry
Profesor Asistente, DCC, Universidad de Chile. PhD Computer Science, Vrije Universiteit Brussel, Bélgica (2005). Master of Science in Computer Science, Vrije Universiteit Brussel, Belgium, Ecole Des Mines de Nantes, Francia y Universidad de Chile, Chile (1999).
jfabry@dcc.uchile.cl

José Miguel Piquer
Profesor Asociado, DCC, Universidad de Chile. Doctor en Computación, École Polytechnique de Paris. Director Técnico de NIC Chile y Director de NIC Labs.
jpiquer@dcc.uchile.cl

1. INTRODUCCIÓN

1.1 Acerca de los juegos y el aprendizaje

¿Han visto una gata traer un ratón vivo a su gatito para que juegue con él? A través de la persecución los cachorros se divierten y aprenden a cazar su comida y las habilidades necesarias para convertirse en adultos.

Si nos fijamos en el proceso vemos que el aprendizaje es divertido y que los humanos no tenemos el monopolio del arte de enseñar a nuestros hijos.

1.2 Aprender haciendo / entendimiento

Algunas habilidades se aprenden mejor "haciendo", mientras que otras simplemente "por la comprensión". El aprender "haciendo" tiene la ventaja de dejar imágenes más vívidas en la memoria, pero puede ser demasiado costoso: se aprende mejor la geografía en los viajes, pero se puede aprender sobre muchos más lugares distintos al leer libros de geografía. "Lo esencial, sin embargo, es que el aprendizaje debe ser un placer para ser eficaz y debe seguir, como Edison lo propuso, 'el instinto natural del ser humano' [Wozniak]".

2. IE2001 TALLER DE PROYECTO

El curso IE2001, Taller de Proyecto, se inició en 2008 con el nuevo Plan de Estudios de la carrera de Ingeniería de la Escuela de Ingeniería y Ciencias de la Universidad de Chile, para permitir a los estudiantes de segundo año 'degustar' la aplicación de un área de su elección.

Los módulos proponen diferentes temas relacionados con las disciplinas de varios departamentos de la Facultad de Ciencias Físicas y Matemáticas (FCFM), destinados a grupos de diversos tamaños con el objetivo común de que los alumnos aprendan habilidades demandadas por la Ingeniería mientras trabajan en un proyecto concreto.

En 2009 los temas de los módulos abarcaban un gran espectro; desde aquel que enseñaba a diseñar sitios en "Second Life" hasta uno que se creaba y evaluaba un biorreactor para una comunidad rural en Chile. En tanto, nuestro Departamento de Ciencias de la Computación (DCC) propuso un módulo, Mindstorms, donde los estudiantes aprendieran a programar robots Lego y, en consecuencia, a conectar el mundo virtual de la Computación con uno más concreto y físico.

2.1 Módulo Mindstorms

Este Módulo surgió como un experimento, después de un retiro estratégico realizado por los profesores del DCC, donde discutimos nuevas formas de enseñar a programar que resultaran más atractivas. Una de éstas, que se ha comenzado a usar en cursos de primer año en algunas universidades, es a través de la programación de robots. El Departamento de Ingeniería Eléctrica contaba con varios kits Mindstorms, programables en Java usando LeJOS (Lego Java Operating System), y decidimos probar si esto resultaba realmente factible y atractivo como un segundo curso, después que los alumnos ya habían aprobado el primer curso de programación Java en primer año.

Fue así que Mindstorms comenzó como un módulo de Seminario de Diseño en 2007 y luego se transformó en Taller de Proyecto en el nuevo Plan de Estudios de Ingeniería en 2008 y 2009.

Inicialmente trabajamos con ocho kits Lego Mindstorms. Y este año compramos diez kits de la nueva versión Mindstorms NXT y comenzamos a utilizar LeJOS NXJ, que es la más reciente versión del software. En los dos primeros años los estudiantes se organizaban en equipos de tres o cuatro y se reunían semanalmente durante tres horas en una sala del edificio de electro-tecnologías de la Facultad. Este año organizamos ocho grupos de cuatro estudiantes para que trabajaran en el laboratorio de nuestro Departamento en cualquier horario. Este hecho mejoró considerablemente el ambiente de trabajo.

En todos los casos los alumnos deben aprender, primero, a enfrentar las dificultades que presenta trabajar con robots: errores de hardware, software, mecánicos, etc. Después de superar exitosamente las dificultades de un primer proyecto común de fácil construcción y programación (un robot capaz de seguir una línea negra demarcada en el suelo), los estudiantes eligen un tema y comienzan a diseñar y construir el robot durante el semestre.

El objetivo del curso es tratar que los alumnos mezclen dos habilidades para desarrollar su proyecto: realizar el diseño y la construcción del robot (más relacionado con la mecánica y la física) y el diseño e implementación del programa que lo controla. Dado que es un curso que dictamos desde el DCC, tratamos que el énfasis mayor del proyecto y su evaluación estén orientados hacia la programación. Pero obviamente son dos aspectos ineludibles al momento de evaluar el éxito del proyecto completo.

Los proyectos (que se describen en la página wiki del curso [https://wiki.dcc.uchile.cl/TallerMindstorms]) son muy diversos. Por ejemplo, en 2009 tuvimos un tanque explorador de caminos de piedras siguiendo el modelo de su primo en Marte; un traductor de Morse capaz de escribir los símbolos recibidos en una pizarra blanca, y muchos otros proyectos. El mejor proyecto de esta instancia del curso, elegido por el grupo docente, fue una innovadora técnica de control remoto (vía Bluetooth) de un auto, sobre la base de captores de ultrasonidos del kit de Lego.

Proyectos desarrollados en 2007:

1. **Robot mascota,** reaccionando al tacto y sonido.
2. **Tirador:** robot lanzador de bolas dirigidas a un objetivo encendido;
3. ***Tracker:** dos robots, uno se mueve al azar con una luz en la parte superior y el otro trata de tocarlo. Una vez que hacen contacto ambos se detienen.
4. **Policía y ladrón:** un robot se roba un "tesoro" y el otro trata de detenerlo.
5. **Que evita obstáculos:** robot que busca una manera de avanzar en un conjunto aleatorio de obstáculos.
6. ***El jugador de póquer:** Un robot "croupier" que distribuye las cartas en una partida que sigue las reglas del juego en función del número de jugadores.
7. **Robot Escritor:** robot con un bolígrafo que intenta escribir una frase moviéndose sobre una pizarra blanca.

(*) elegidos mejores proyectos.

2008:

1. ***Robot que sube escaleras:** construcción para subir escaleras de tamaño real.
2. **Luchador de sumo:** robot que trata de sacar un objeto desde un círculo.
3. **Jugadores de pelota:** dos robots se comunican a través de infrarrojos para coordinar un juego.
4. **Bailarines:** dos robots que coordinan sus movimientos a través de infrarrojos.
5. ***Solucionador de laberinto:** robot que navega en un laberinto estructurado.
6. **Lector de Morse:** robot que lee un código con la ruta para recorrer un laberinto y la utiliza para atravesarlo.
7. **Parking robot:** robot coche que intenta estacionarse automáticamente entre dos paredes.

(*) elegidos mejores proyectos.

2009:

1. **Tanque explorador**, de caminos peligrosos basado en el modelo de su primo en Marte.
2. ***Traductor Morse** capaz de escribir los símbolos recibidos en una pizarra.
3. **Programa de solución** cubo de Rubik.
4. **Robot** que instala un puente para cruzar abismos.
5. **Perro caminador.**
6. **Robot que recorre un laberinto.**
7. **Robot de exploración** que construye un mapa del lugar.
8. *Innovadora **técnica de control remoto** basada en los captores de ultrasonidos de los kits de Lego.

(*) elegidos mejores proyectos).

2.2 Futuro del Módulo

El módulo se ha ofrecido en tres ocasiones y está evolucionando cada vez más (ver recuadros). Las principales lecciones aprendidas han ido influenciando las versiones siguientes. Por ejemplo, descubrimos que un buen porcentaje de nuestros alumnos jamás jugó con Legos, por lo que les enseñamos a armar el primer robot. También vimos que tienden a dejar la programación para el final, lo que muchas veces los llevaba a batallar con el robot todo el semestre, programándolo muy mal en los últimos días (ahora les pedimos ir mostrando el código durante el semestre). Asimismo nos dimos cuenta que muchos proyectos no dejaban documentación que permitiera replicarlos, y al desarmar el robot al final de semestre perdíamos información valiosa.

En el próximo módulo planeamos proveerle a nuestros estudiantes herramientas para que graben en video sus proyectos y los publiquen en medios de comunicación públicos como YouTube; los animaremos además a reimplementar proyectos anteriores del curso con el fin de que puedan potenciarlos, de modo que estos pasen a ser el resultado de un verdadero trabajo en equipo entre varios grupos e iteraciones del curso. Esto, de manera similar a los proyectos de ingeniería del mundo real (podemos imaginar las catedrales de la vieja Europa desarrolladas a lo largo de varias generaciones de arquitectos, en particular la de La Sagrada Familia de Barcelona).

Características del curso actual

- Un sitio wiki (iniciado por los auxiliares y completado por los estudiantes) con documentación completa y ejemplos de proyectos en español, incluyendo instrucciones sobre cómo construir y programar varios proyectos de robots, y fotos y videos de los robots en acción.
- Manejamos la última versión de los kits de Mindstorms, NXT, donde el hardware es más fiable y fácil de programar, usando la plataforma LeJOS, kit abierto de programación de código fuente en Java.
- Los estudiantes tienen acceso permanente al laboratorio del DCC para que también trabajen en sus robots fuera de las horas de clases (a menudo se ven algunos grupos construyendo sus robots incluso durante un viernes en la noche).
- Potenciamos el enfoque en la documentación del proyecto (que debe permitir que otro grupo de estudiantes pueda reconstruir totalmente y reprogramar el robot) en contraste con el robot mismo (que debe ser "destruido" al final del semestre).

Características del curso futuro

- Haremos dos secciones para aumentar el impacto y el número de proyectos.
- Mejoraremos los medios para publicar los videos de los proyectos (por ejemplo, en YouTube);
- Elaboraremos un índice de los videos ya disponibles en todo el mundo;
- Mejoraremos la reutilización de proyectos pasados, la obtención de progresos y un mejor incentivo para ser claro en la documentación;
- Dispondremos de nuevas herramientas para los estudiantes, tales como Bluetooth de conexión entre un robot LEGO y un computador;
- Enseñaremos el valor de la ética y la necesidad de documentar cada referencia externa que inspiran el proyecto, en particular la prevención de acusaciones de plagio;
- Adicionalmente enseñaremos sobre el componente de la Ingeniería del módulo: si algo no está funcionando (sobre todo si es un componente externo), ¡arréglalo o sustitúyelo por otra cosa que sí funcione!

2.3 Opiniones de los Estudiantes

El curso es altamente valorado por los estudiantes y esperamos duplicar su oferta el próximo año. Los estudiantes también están pidiendo cursos similares

en los niveles superiores del programa de estudios universitarios. E incluso algunos piden ser auxiliares de la próxima entrega del curso, de modo de tener la oportunidad de trabajar más con los kits de Lego. Haber traído Mindstorms al laboratorio del DCC provocó una especie de "envidia" sana de los alumnos mayores de nuestro Departamento, que observan por las ventanas cómo los jóvenes de este curso se divierten armando y programando Legos.

En los próximos años trataremos que los estudiantes continúen trabajando con los kits de Lego también durante el segundo semestre, cuando nadie los ocupa. Por ejemplo, podrán trabajar en algún proyecto de investigación en colaboración con un profesor.

3. CONCLUSIÓN

3.1 Una nueva era para el aprendizaje

Algunos temas, como la energía nuclear y la cirugía, probablemente habrá que aprenderlos por muchos años más en un entorno virtual y teórico, ya que sería demasiado costoso o peligroso dejar que los estudiantes experimenten con el material real sin antes tener una completa comprensión de los aspectos teóricos detrás de la práctica. Sin embargo, la tecnología ha reducido el costo del aprendizaje de muchos otros temas en condiciones casi reales, tales como la geografía y los idiomas: la comunicación ha disminuido tanto sus costos que los estudiantes actuales pueden oír y ver personas y países extranjeros, aun cuando viajar allí no sea factible, por seguridad, tiempo o razones ecológicas.

Pero la verdadera revolución de nuestra era será la enseñanza de las tecnologías de la información, donde los estudiantes pueden tener la plena experiencia práctica de los temas que aprenden, a un precio cada vez más viable:

- El acceso a computadores se limitaba a una elite de científicos en los años '60. Ahora están disponibles para la mayoría de la gente en los países desarrollados y en vías de desarrollo. La enseñanza de la programación informática ha sido tradicionalmente limitada a un entorno simulado cerrado (sin aplicaciones reales). Pero ahora se ofrecen algunos cursos para programar aplicaciones reales para el teléfono celular del estudiante [iPhoneClassProjects], o incluso para construir un computador con sus componentes electrónicos y software, incluido un editor de texto y un video juego [NANDtoTetris].
- La experimentación con robots se limitaba a los científicos y la capa superior de los ingenieros de la NASA. Ahora ha sido puesta a disposición del público en general a través de diversas actividades (el lenguaje "Logo" de la tortuga, el "Bigtrack" de tanques programables, y el mismo Mindstorms de Lego).
- El costo de la impresión 3D ha disminuido al punto que los estudiantes del MIT la utilizan en sus cursos para pequeños prototipos y diseños de Ingeniería general. Y el "Fab Labs" se instala en los países en vías de desarrollo para ayudar a la enseñanza de esta área a los estudiantes locales [FabLabs].

Como el acceso a la tecnología de alto nivel se está democratizando más rápido y de forma más profunda que nunca, ya no hay excusa para seguir haciendo cursos tradicionales donde los estudiantes aprenden habilidades en una limitada forma teórica antes de salir al mundo del trabajo. Al contrario. Los estudiantes requieren "aprender a aprender" a través de su propia experiencia y razonamiento, ya que necesitarán seguir aprendiendo por mucho tiempo después de salir de la Universidad, en un mundo donde el cambio tecnológico ha superado el ritmo de las generaciones humanas. "Taller de Proyecto" es sólo el comienzo de una nueva generación de cursos y escuelas. BITS

REFERENCIAS:

1) [iPhoneClassProjects] Where Phones in Class Are OK [http://www.insidehighered.com/news/2009/08/20/iphone]

2) [NANDtoTetris] From NAND to Tetris Building a Modern Computer from First Principles By Noam Nisan and Shimon Schocken [http://www1.idc.ac.il/tecs/] [http://mines.humanoriented.com/proposals/nand-to-tetris/] [http://video.google.ca/videoplay?docid=7654043762021156507&ei=FyQkSob8OI-IrwKzkaCQAg&q=techtalks]

3) [Wozniak] small story about Edison and Tesla [http://www.supermemo.com/articles/genius.htm#Edison]

4) [FabLabs] Neil Gershenfeld on Fab Labs [http://www.ted.com/talks/lang/eng/neil_gershenfeld_on_fab_labs.html] [http://fab.cba.mit.edu/]

EDUCACIÓN EN INFORMÁTICA

Enseñar y Aprender HCI:
Amalgama Perfecta entre Teoría y Praxis

Jaime Sánchez Ilabaca
Profesor Asociado, DCC, Universidad de Chile. Doctor en Informática y Educación, Columbia University, Nueva York. Director del Centro de Computación y Comunicación para la Construcción del Conocimiento (C5), DCC, Universidad de Chile.
jsanchez@dcc.uchile.cl

Mauricio Sáenz Correa
Investigador del Centro de Computación y Comunicación para la Construcción del Conocimiento (C5), DCC, Universidad de Chile. Ingeniero Civil en Computación. Alumno de Magíster en Ciencias mención Computación, en la misma Universidad.
msaenz@c5.cl

INTRODUCCIÓN

A través de los años, el aprendizaje ha sido explicado por medio del análisis de diversas teorías que postulan variados modelos, principalmente enfocados en distinguir diferentes grados de actividad/pasividad en el rol del estudiante (Sánchez, 2001).

La enseñanza tradicional o frontal ubica a los alumnos frente a un profesor que provee teorías, conceptos y contenidos diversos. En este escenario, los alumnos son meros receptores pasivos de la información que presenta y explica un profesor transmisor ya sea con la ayuda de tecnología añeja (pizarrón) o tecnología digital (proyector, computador). Esta forma de enseñar y aprender, que responde a un modelo conductista-positivista, ha prevalecido por décadas en la educación y lentamente experimenta cambios e innovaciones que apuntan hacia un modelo donde el aprendiz es el actor principal y activo de su aprender, construyendo conocimiento a través del cambio en el significado de la experiencia, la interacción con otros y la organización y reorganización de sus esquemas mentales.

En este nuevo escenario el rol del profesor es de facilitador de experiencias significativas de los alumnos y mediador o coach de la construcción de conocimiento de los aprendices por medio de la práctica, la discusión, el análisis, la comprensión y el compromiso activo.

Es precisamente en este último contexto en el que se desenvuelve la enseñanza y aprendizaje de la Interacción Humano-Computador (HCI), en la carrera de Ingeniería Civil en Computación y los programas de posgrado en Ciencia de la Computación de la Facultad de Ciencias Físicas y Matemáticas de la Universidad de Chile. Este curso de HCI ha sido pionero en Chile y Latinoamérica y se imparte anualmente de manera ininterrumpida por más de 20 años, innovando y actualizándose constantemente en sintonía con la tecnología de punta imperante. Como resultado, los alumnos han logrado tomar conciencia de un amplio rango de situaciones generales de la Interacción Humano-Computador,

que deben ser consideradas al diseñar cualquier tipo de software o dispositivo para el uso de personas. Desde el punto de vista del diseño, ello implica desarrollar una capacidad para analizar problemas sobre una base técnica, cognitiva y funcional, y analizar críticamente métodos y técnicas de diseño interactivo de Interfaces.

La principal razón por la cual se dejan de utilizar muchos sistemas radica en que las interfaces no son concebidas y diseñadas para que los usuarios finales interactúen con ellas, proveyendo falsas pistas para su uso y mecanismos que inducen al error. El objetivo de la HCI es generar productos usables, entendibles, seguros, funcionales, inteligibles y utilizables. Productos cuyas interfaces sean fáciles de interpretar, entender, que provean pistas visibles para su operatividad y consideren el error humano en su diseño.

La HCI se preocupa de que los usuarios finales sean el centro de cualquier diseño y desarrollo. Utiliza el diseño centrado en el usuario, esto es, el diseño de la aplicación nace desde las necesidades e intereses del usuario, transcurre con la participación permanente del usuario y culmina con su usabilidad. Por ello la interacción humano-computador propicia sistemas cuyas interfaces sean diseñadas a partir de las necesidades del usuario, conociéndolo, entendiéndolo e integrándolo al equipo de diseño. Con esto se asegura que la aplicación sea usable y entendible y, en consecuencia, utilizada por el usuario final. Al mismo tiempo, estos sistemas deben modelarse e implementarse considerando todas las especificaciones técnicas y funcionales y siguiendo metodologías de ingeniería de software, de manera tal que se obtenga un sistema robusto, funcional y confiable (Sears & Jacko, 2009). La idea es generar aplicaciones cuyas interfaces ayuden, mejoren y amplíen el ámbito de las experiencias del usuario, junto con entregar soluciones más confiables ahorrando en costos, optimizando la eficiencia de procesos y, en definitiva, mejorando la productividad, eficiencia y la vida de las personas.

La Interacción Humano-Computador, además de estar asociada a la ingeniería de software, está fuertemente influenciada por otras disciplinas del conocimiento tales como las ciencias cognitivas, la biología, psicología, sociología, antropología, el diseño gráfico e industrial y la ergonomía. Todas estas disciplinas contribuyen e informan al diseño de la interacción humano-computador respondiendo a ciertas interrogantes: ¿cómo se comporta el ser humano en contextos sociales y culturales en el tiempo y espacio?, ¿cómo ocurren sus relaciones e interacciones sociales?, ¿cómo perciben los usuarios, memorizan, procesan la información, conocen, aprenden, resuelven problemas?, ¿cómo el cerebro y los órganos de los sentidos participan en la interacción con software y dispositivos digitales?, ¿cómo los conceptos, teorías, elementos y componentes del diseño gráfico e industrial son incorporados en el diseño de las interfaces usuarias?

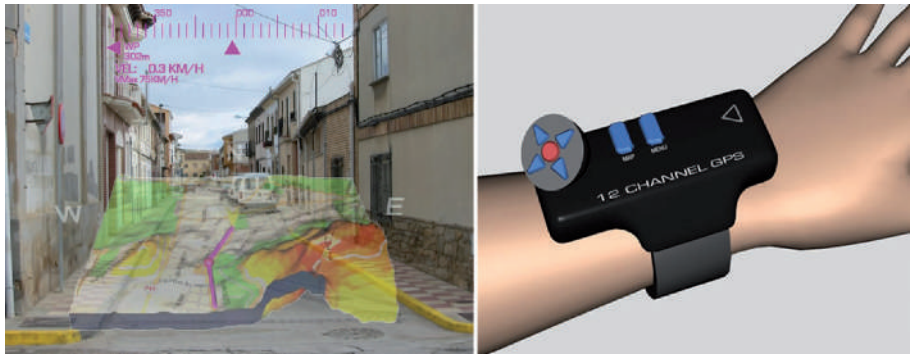


Figura 1. Propuestas de interfaces para dispositivos GPS de alumnos del curso del año 2006.

TEORÍA Y PRÁCTICA

Para Myers (1998) es importante que los alumnos de las carreras en Computación e Informática conozcan y aprendan acerca de las temáticas que subyacen las interfaces usuarias. El autor señala que las interfaces son el valor agregado de los sistemas computacionales y que generarán diferencias competitivas, ya que el hardware y el software pasarían a ser commodities. Su raciocinio se basa en que si los alumnos no saben acerca de la interacción humano-computador, no serán útiles para las necesidades de la industria. La realidad ha coincidido con estos planteamientos. La gran competencia comercial de los productos tecnológicos actuales gira en torno a lo novedoso, interactivo, usable y útil de las interfaces. Así por ejemplo, la actual competencia comercial en el campo de las interfaces *touch* y sus variados diseños, apunta a interfaces cada vez más naturales, usables y entendibles para los usuarios.

Una de las claves del curso HCI ha sido la metodología de aprendizaje y la innovación y actualización permanente. Esta metodología ha consistido en aplicar prontamente las teorías, conceptos y modelos estudiados en clases a situaciones de trabajos prácticos, aplicados, reales y atingentes. Sobre la base de un tema global tratado durante todo el curso, los alumnos resuelven y desarrollan una serie de casos de aplicación de HCI y un proyecto final de diseño de la interacción humano-computador, que integra y unifica los resultados de los casos previamente desarrollados, escalando en requerimientos, profundidad y complejidad.

Así como hoy una tendencia es que las interfaces giren en torno a los dispositivos *touch*, los cambios tecnológicos son tan rápidos y diversos que constantemente están surgiendo nuevas y variadas interfaces. La disciplina de la Interacción Humano-Computador es especialmente sensible a los cambios e innovaciones permanentes de la tecnología en el mundo. De esta manera el aprendizaje de la disciplina implica que, además de asistir a clases, construir conocimiento y manejar información, el alumno debe leer constantemente *papers* de las últimas revistas y conferencias del área para conocer tendencias y temas actuales de HCI.

Las clases del curso de HCI se imparten con diversos matices y orientaciones. Algunas sesiones consisten en presentar, analizar y discutir teorías, modelos y contenidos, mientras otras consideran trabajo colaborativo en equipo, con planificación, desarrollo y evaluación del avance en el diseño de los casos de aplicación. Periódicamente los

alumnos revisan y analizan sus progresos con el cuerpo docente, quienes apoyan, median y facilitan las decisiones conceptuales y funcionales que deben tomar estos.

CASOS DE APLICACIÓN

Como se ha dicho, el curso plantea a los alumnos casos de aplicación de diseño o rediseño de la interacción humano-computador que deben resolver. Su trabajo evoluciona desde el diseño de interfaces humano-máquina al diseño de interfaces humano-computador. Estos casos son organizados en un tema central tal como transporte, educación y navegación, concatenándolos para lograr un diseño de HCI más completo y materializado en un proyecto final.

Todas las propuestas de resolución de casos de aplicación de cada equipo son presentadas, analizadas y discutidas en el curso. Esto implica que además de aprender conceptos y su aplicación, los alumnos adquieren y refuerzan habilidades para exponer ante una audiencia. A medida que el curso avanza y los casos de aplicación se complejizan, las exigencias sobre la calidad de la presentación aumentan, lo que los obliga a exponer, aplicar y utilizar prolija e impecablemente los contenidos estudiados.

La temática de los casos de aplicación es variada. Los alumnos han diseñado y rediseñado diversos sitios y portales Web, dispositivos de audio e información para vehículos, controles remoto, equipos multifuncionales de oficina, sistemas de control a distancia del hogar, sistemas de control asistido de cárceles, zoológicos y centros comerciales, navegación con dispositivos GPS (Figura 1), sistemas de información para el Transantiago (buses de la capital) y aquellos ligados a la educación, entre otros. Uno de los puntos claves de estos estudios es que los diseños deben estar centrados en el usuario, lo que implica trabajar con y para los usuarios finales, quienes evalúan prototipos de diseño y aportan nuevas visiones, problemas, inquietudes, intereses y necesidades.

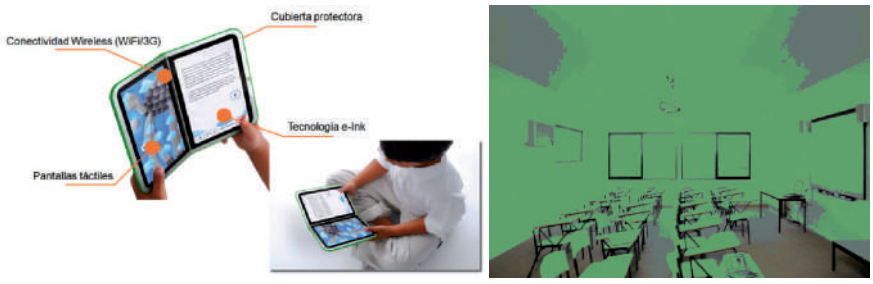
Los casos de aplicación de HCI no sólo son variados sino que también atingentes a situaciones reales en contextos reales. Así por ejemplo, en 2001 se realizó un análisis de las interfaces de los nuevos cobradores automáticos instalados en los ya desaparecidos 'micros' (buses) amarillos de Santiago. En el año 2007 se estudió el diseño de una aplicación que permitiera al usuario ubicar la ruta más óptima que lo llevase a su trabajo/hogar, además de la mejor opción de medio de transporte. En 2008 el tema del curso fue el Transantiago (Figura 2A). Los alumnos evaluaron y rediseñaron el portal de Transantiago (www.transantiago.cl) (Figura 2B). El proyecto final sistematizó y reunió todos los casos de aplicación antes resueltos, pero con un mayor grado de complejidad, como el diseño de un sistema-solución de transporte enfocado principalmente a proveer información para ayudar a la toma de decisiones, considerando la comodidad de los pasajeros, el contexto y su entorno. Para esto los alumnos tuvieron que considerar que la interfaz propuesta debía ser implantada en paraderos de micros y estaciones de metro, además de otros lugares que los alumnos estimaran conveniente (Figura 3).

Este año el tema central del curso fue educación. Bajo esta temática, los alumnos diseñaron la interfaz innovadora de un dispositivo que sirviera como texto digital para alumnos de un colegio. Para ello los alumnos debían buscar una solución tecnológica que permitiese comportarse como un libro, una estación de trabajo o un sistema de búsqueda de información. Esto según los requerimientos del usuario, de manera de presentarse como un recurso reutilizable de bajo costo de mantención y que permitiese la actualización de contenidos. Al problema práctico expuesto se agregaba una variable tecnológica importante, que el libro impreso aún no ha sido reemplazado por ninguna de las tecnologías digitales existentes, por lo que el dispositivo debía considerar una correcta funcionalidad basado en un diseño creativo e innovador. Como resultado, los alumnos generaron propuestas de interfaces interesantes e innovadoras utilizando, en algunos casos, dispositivos de hardware existentes y, en otros, generando propuestas propias de hardware (Figura 4).

El segundo caso de aplicación de este año consistió en que los alumnos evaluaran las interfaces del portal educativo EducarChile (www.educarchile.cl), uno de los portales Web públicos más visitados y utilizados a nivel nacional. Luego de esta evaluación los alumnos rediseñaron el portal en su proyecto final (pensando en los usuarios finales: profesores, alumnos y apoderados). Ya concluido, el trabajo fue presentado a un grupo de profesionales responsables del portal en la Fundación Chile y el Centro de Educación y Tecnología que coordina la Red Enlaces (www.enlaces.cl) (Figura 5). Como resultado de esta experiencia los

Figura 2. (A) Diseño de una aplicación móvil de apoyo al transporte público. (B) Propuesta de diseño móvil para el sitio Web de Transantiago.

Figura 3. Interfaces del sistema diseñado por los alumnos para el sistema-solución de un paradero del Transantiago.



(A)

(B)

(C)

Figura 4. (A) Interfaces de libro digital diseñada por un equipo de alumnos del curso. Los alumnos no sólo se preocupan de diseñar la interfaz o el dispositivo de hardware, sino también consideran el contexto en que será utilizada la tecnología que están diseñando. **(B)** Rediseño de la interfaz del portal EducarChile. **(C)** Propuesta de Interfaz móvil para el portal EducarChile.

gestores del portal rediseñaron el sitio para hacerlo más usable, entendible y funcional. Así los profesionales del portal generaron una encuesta para obtener más datos y opiniones de sus usuarios finales (http://www.educarchile.cl/Portal.Base/Web/VerContenido.aspx?GUID=c363e908-c2a3-4b21-bd56-ad0cd6ad6d94&ID=197268).

EVALUACIÓN

La evaluación de los principales conceptos estudiados en clases se realiza mediante el uso de la técnica de los mapas conceptuales (Figura 6). De esta manera la evaluación no se centra en cuánto recuerdan o memorizan los alumnos, sino que en cómo entienden, ordenan, jerarquizan, asocian y relacionan los conceptos aprendidos para construir significado. Así es posible conocer la representación de la casa conceptual mental de los alumnos sobre la base de lo aprendido. Cada alumno culmina el curso con un mapa conceptual de los principales conceptos adquiridos y sus asociaciones, esto es, una representación gráfica de su aprendizaje significativo de conceptos de HCI.

OPINIONES DE LOS EX - ALUMNOS

Interesantes e ilustrativas son las opiniones de algunos alumnos del curso, varios de ellos hoy egresados, que han tomado el curso de Interacción Humano-Computador. Con relación a la metodología de aprendizaje, los alumnos opinan que es una manera distinta y que apoya fuertemente el aprendizaje de los diversos conceptos vistos en el curso, "Fue todo un gusto y privilegio haber podido aprender de esta manera" (Víctor Toledo, ex-alumno de 2009); "El curso de HCI me pareció interesante en el sentido de que se estudian casos reales como proyecto del curso. Además permite que uno como estudiante de rienda suelta a su creatividad e imagine soluciones que son innovadoras e interesantes" (Juan Pablo Rodríguez, ex-alumno de 2008). Asimismo los estudiantes reconocen que la metodología utilizada permite apoyar el desarrollo de otras habilidades que son fundamentales para los profesionales de hoy: "El curso de Interfaces Humano-Computador es muy interesante, ya que permite adquirir y desarrollar habilidades que no son comunes en otros cursos del DCC [...] los constantes debates que se plantean y las presentaciones del curso me permitieron potenciar mis habilidades de comunicación oral, las cuales en el mundo laboral son muy valoradas." (Angelo Tadres, ex-alumno de 2007).

En términos más globales, acerca de la aplicabilidad del curso, las opiniones coinciden en que otorga herramientas útiles para los ingenieros que hoy egresan. Angelo Tadres dice: "Como futuros ingenieros constantemente se nos enseña a diseñar software funcional y de calidad. Pero nunca se nos pide que pensemos en los usuarios de dichas soluciones. El curso me entregó las herramientas necesarias para resolver estos problemas, creando así interfaces mucho más usables y visualmente atractivas".

Otras opiniones destacan lo aprendido en el curso como algo útil para sus desempeños profesionales actuales: "El curso de HCI dictado por el DCC ofrece a sus alumnos una visión distinta de la ingeniería de software, donde la experiencia de uso del software tiene tanta importancia como la calidad técnica de su desarrollo [..] En mi caso, me

Figura 5. Presentación del rediseño del portal EducarChile.cl a profesionales de Fundación Chile y la red Enlaces del Mineduc.

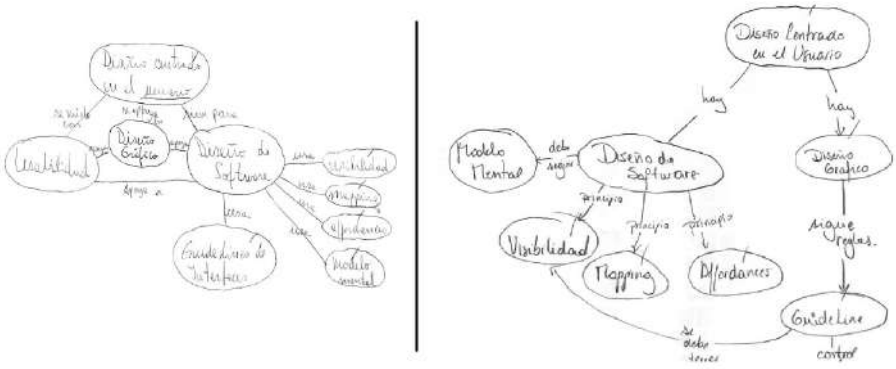


Figura 6. Mapas conceptuales generados por alumnos del curso HCI en función de los conceptos analizados en clases.

ha tocado estar a cargo de desarrollos de aplicaciones Web cuya usabilidad debe ser tomada muy en cuenta, ya que un mal diseño de sus interfaces e interacciones provoca una merma en la productividad de los sistemas a la hora de su utilización" (Claudio Oyarzún, ex-alumno de 2004); "El curso de interfaces nos enseña a concentrarnos en lo que finalmente utilizará el usuario, la interfaz del software, punto débil en muchos software que he tenido que usar" (Mauricio Zúñiga, ex- alumno de 2004); "[..] la sensibilidad HCI puede ser siempre una herramienta favorable, un 'plus' dentro de las cualidades de un Ingeniero de Desarrollo...y que hoy en día es escasa y por tanto valorada" (Miguel Elías, ex-alumno de 2005); "Hasta antes de hacer este curso estaba acostumbrado a desarrollar programas orientados a sistemas, concentrándome sólo en que el programa funcione. Este fue el primer curso de la especialidad en que el foco estaba lejos de programar, sino que estaba en la experiencia de uso de un usuario final; usuario que no debe ser necesariamente un experto en Computación. [..] En mi trabajo desarrollo muchas aplicaciones Web y este ramo me ha servido enormemente para discutir con otros programadores que les cuesta ponerse en el lugar del usuario final y tienden a diseñar para ellos mismos" (Thomas Pieper, ex-alumno de 2002).

Estas opiniones reflejan lo relevante que ha sido para estos profesionales haber estudiado la Interacción Humano-Computador bajo un enfoque de aprendizaje constructivista, con casos de aplicación y proyectos de diseño de interfaces de software de manera transversal a lo largo del curso.

Incluso hay algunos estudiantes que no sólo adoptan a su campo laboral el conocimiento y la experiencia aprendida en el curso de HCI, sino que lo llevan a un plano más personal, con una mayor sensibilidad por los temas de interfaces humano-computador: "Al final del curso tenía ya interiorizado el análisis de interfaces en mi vida diaria, no solamente en las páginas Web que diariamente visito, sino que también en aparatos como reproductores MP3 y celulares que uso... algo que he seguido aplicando consciente e inconscientemente hasta el día de hoy" (Miguel Elías, ex-alumno de 2005).

CONCLUSIONES

Este texto presenta una forma activa de enseñar y aprender la interacción humano-computador para alumnos de las carreras y programas de posgrado en Computación e Informática. La metodología de aprendizaje del curso de HCI que se imparte en el Departamento de Ciencias de la Computación (DCC) de la Universidad de Chile permite un trabajo activo, amplio, concreto y actualizado que lleva a los alumnos a construir conocimiento desde la práctica. El curso requiere que los estudiantes apliquen constantemente los conocimientos aprendidos en clases. Como resultado, existe una apropiación más profunda de la teoría y práctica de la Interacción Humano-Computador.

Nuestros usuarios finales son los alumnos. Y sus reacciones y comentarios respaldan el hecho de que el enfoque de la metodología de aprendizaje esté más centrado en el trabajo práctico de aplicación que en la mera presentación y análisis de contenidos, así como también resaltando cómo esto ayuda a la transferencia de dicho conocimiento a otros ámbitos de sus vidas.

La relevancia de un curso de HCI en la formación de un ingeniero en Computación radica en que otorga otra mirada a los sistemas y soluciones que cotidianamente diseñan y desarrollan estos profesionales. Al empotrar conceptos, ideas, teorías y modelos de HCI puede existir mayor certeza que sus desarrollos serán utilizados y entendidos por los usuarios finales y, por tanto, su trabajo tendrá mayor valor, solidez y sensibilidad para estos.

La historia está plagada de ejemplos de aplicaciones que su uso se ha visto impedido porque sus interfaces no son usables, accesibles, entendibles o no representan los modos de interacción de los usuarios finales. Un curso de HCI puede hacer la diferencia.BITS

REFERENCIAS

Sánchez, J. (2001) Aprendizaje Visible, Tecnología Invisible. Dolmen Ediciones.

Sears, A., Jacko, J. (2009) Human-Computer Interaction, Development Process. CRC Press, Taylor & Francis Group.

Myers, B. (1998) A Brief History of Human Computer Interaction Technology. ACM Interactions, 5(2), pp. 44-54.

Entrevista Internacional

Claudia Bauzer Medeiros

Por Alejandro Hevia

Cláudia Bauzer Medeiros es docente y directora del Laboratorio de Sistemas de Información del Instituto de Computación de Universidade Estadual de Campinas (UNICAMP), Brasil. Es miembro de los comités de Ciencia de la Computación e Ingeniería de la Fundação de Amparo à Pesquisa do Estado de São Paulo, y de Ciencia de la Computación de la Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES). Asimismo es embajadora del Comité de Mujeres en Computación de la ACM. Fue presidenta de la Sociedad Brasileña de Computación (SBC) entre 2003 y 2007, y actualmente integra el consejo directivo de dicha Sociedad. La SBC es una de las sociedades en su tipo más grandes de Sudamérica; fundada en 1978, en la actualidad está integrada por cerca de siete mil socios, entre estudiantes, profesionales, docentes e investigadores del área, además de los socios institucionales como universidades y empresas nacionales y transnacionales.

¿Cómo piensas que la Ciencia de la Computación (CC) debería ser enseñada en las universidades?

Realmente depende del tipo de cursos y habilidades que quieras enseñarles a los estudiantes. Una base teórica fuerte asociada con la resolución de problemas de la vida real, parece ser la mejor mezcla.

Para poder avalar mejor mi respuesta, daré un par de datos acerca de Brasil. Existen aproximadamente 1.900 cursos relacionados con la CC en Brasil, con 370.000 estudiantes inscritos y 38.000 graduados por año (estadística del 2007: https://www.sbc.org.br/educacao/lib/exe/fetch.php?media=documentos:estatisticas-2007.pdf). Esto corresponde a un porcentaje muy pequeño de la población total de estudiantes universitarios del país. Hay incontables tipos de cursos ofrecidos -ingeniería en Computación es el único que dura cinco años, y el resto dura o cuatro años, ofreciendo grados en CC, sistemas de información, análisis de sistemas, ingeniería de software, etc; o 3 años en grados univeristarios a nivel técnico-. Todas estas carreras cubren algunos temas de la Ciencia de la Computación a un mayor o menor nivel de profundidad, dependiendo de los objetivos -un programa de 4 años en Ingeniería de Software cubrirá temas básicos en estructuras de datos y algortimos, pero se concentrará mucho más en temas de

software-. Existen también varias carreras que, aunque no centradas en la CC, ofrecen hasta un 50 por ciento de sus cursos en temas de tecnologías de la información (por ejemplo, en administración de información o ciencia de la información). Finalmente, hay carreas universitarias de cuatro años que están diseñadas para entrenar a los profesores de Computación que ejercen en ciertas licenciaturas.

Hace unos 15 años, la Sociedad Brasileña de Computación (SBC) estableció un conjunto de directrices para ser seguidad por los distintos tipos de carreras universitarias, indicando los temas obligatorios y opcionales y sugiriendo libros de texto. Estas directrices, que son continuamente renovadas, son seguidas básicamente por todas las carreras de CC en Brasil. La sociedad realiza varios talleres para profesores universitarios y, una vez al año, una conferencia de dos días concerniente a los temas de currículo y entrenamiento de profesores. A esta conferencia asisten los coordinadores de carreras de todo el país, ayudando a establecer las políticas de educación en Informática en Brasil. Por lo tanto, nuestra realidad nacional, acerca de cursos y temas, es construida por la comunidad misma de CC, bajo la coordinación de la SBC.

¿Debería enseñarse Ciencia de la Computación más tempranamente, digamos en los colegios?

Sí. La Computación, y el pensamiento computacional, es realmente otro lenguaje y es básico para la sobrevivencia en la sociedad moderna, junto con la lectura, escritura y matemáticas. Este es un aspecto muy importante para atraer más gente joven a las carreras de CC. La mayoría de las escuelas privadas de Brasil tienen cursos básicos de programación. Sin embargo esto no ocurre en el sistema público, donde tales cursos son raros. Estas iniciativas son obstaculizadas por varios factores en nuestro país:

1. La percepción de que cualquiera puede aprender CC solo y, por tanto, los cursos no son necesarios (esto se debe principalmente por la diseminación de herramientas Web en los hogares).
2. Un énfasis reciente en las humanidades y ciencias sociales en las escuelas secundarias está llevando a disminuir las horas de enseñanza de matemáticas y física, por ejemplo, y a aumentar las dedicadas a filosofía y ética.
3. La carencia de carreras universitarias que capaciten a profesores en CC para trabajar en escuelas secundarias.
4. La generalizada disminución del interés de los jóvenes por seguir carreras científicas.

Debo acentuar que los items (1) y (4) son un fenómeno que está ocurriendo en todo el mundo, mientras (2) y (3) son probablemente específicos a Brasil. Por tanto, aunque la mayoría de mis colegas creen que la Ciencia de la Computación debería ser enseñada más tempranamente, existen muchas barreras políticas, culturales y financieras a vencer. La Academia de Ciencias Brasileña está muy preocupada del item (4), y ha lanzado un programa ambicioso para tratar de revertir la situación. Pero esta es una perspectiva de largo plazo y sólo afectará a la Ciencia de la Computación indirectamente.

Existe mucha gente que sostiene que los currículum en Ciencia de la Computación debieran ser modificados, dado que el número de estudiantes que la elige como carrera universitaria está disminuyendo.

¿Qué piensas acerca de eso y qué tipo de modificación crees que podría ser deseable?

Esta es una buena idea que está siendo probada en varios países, incluyendo EE.UU., con la introducción de los currículos multidisciplinarios (como Georgia Tech, Carnegie Mellon, MIT). En Brasil también se está intentando de una forma más tímida. Un buen ejemplo es la aparición en algunas universidades de carreras de cuatro años de duración en Bioinformática. Mientras áreas de cuatro años en Matemáticas Aplicadas son también a menudo actualizadas para ofrecer grados en Matemática Computacional. En la mayoría de los casos sólo unos pocos créditos son dedicados a la interacción con otras áreas (notablemente ciencias médicas y física).

El mayor problema a la hora de crear nuevas carreas es la estructura de regulación impuesta por el Ministerio de Educación de Brasil. La mayor parte de los currículos son regulados por ley y es muy difícil crear nuevas carreras, ya que éstas no tendrán ningún valor si no son aprobadas por las autoridades gubernamentales apropiadas. También la acreditación es renovada periódicamente y las carreras podrían perderla si no se adecúan a las reglas establecidas.

Además la modificación curricular es en sí misma un desafío. ¿Cómo proveer los fundamentos y al mismo tiempo actualizar los contenidos para que los alumnos entiendan la relevancia de lo que están aprendiendo? Las actualizaciones son sólo parte del proceso. Los profesores deben saber cómo presentar los contenidos de una manera integrada con otros, de tal forma que los estudiantes aprendan mejor y más rápido. Pero los profesores capaces de hacer esto, o dispuestos a hacerlo, no son la norma. Y por supuesto, ese es un

tema que trasciende las modificaciones curriculares en el área.

¿Qué sucede con las políticas en Brasil, y en Unicamp en particular, para atraer nuevos alumnos a estudiar Ciencia de la Computación?

No existe estrategia a nivel país, porque el gobierno no está oficialmente conciente del problema. Los grados universitarios y el número de inscritos está creciendo constantemente, y por tanto la disminución de los inscritos en CC no es considerada, desafortunadamente, un problema.

Mi universidad, Unicamp, no tiene una política específica para atraer estudiantes a CC. Sí tiene políticas muy exitosas para atraer buenos alumnos a la Universidad. Una de esas iniciativas es llamada "Universidad de Puertas Abiertas" (UPA), en la cual por dos días se reciben alumnos en su último año de colegio de práticamente todo el país, para mostrarles nuestros departamentos, con demos, entrevistas con profesores, etc. En el último año más de 60.000 jóvenes visitaron Unicamp durante la UPA, y algunos viajaron hasta 1.500 kms para hacerlo. Esta es probablemente la iniciativa más exitosa para atraer estudiantes a la Universidad. Y nosotros también, en CC, captamos una parte de los jóvenes que ven nuestros demos y hablan con los profesores de Ciencia de la Computación, y que prefieren venir hasta aquí antes que a otras universidades. Hasta donde yo sé, ésta es la única universidad en Brasil que tiene un programa nacional en esta dirección.

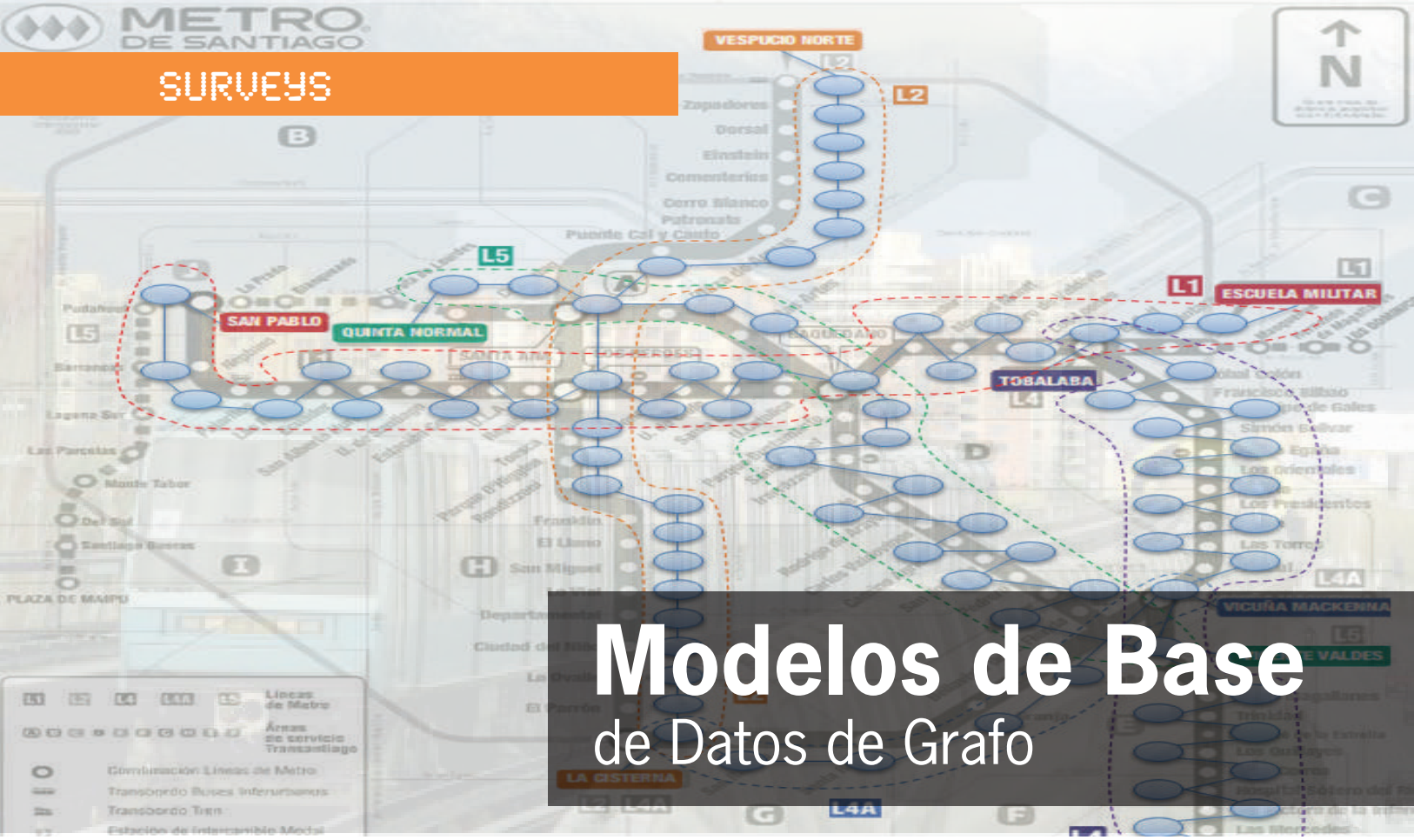
También tenemos muchos tipos de cooperación con otros países, para intercambio de alumnos de CC entre Unicamp y otras universidades específicas, como de Francia, EE.UU. y Argentina. Dentro de estos intercambios los alumnos pueden estudiar desde seis meses hasta un año en otras universidades y obtener créditos de curso por esto. No sé si tenemos este tipo de convenios con Chile. Pero sin duda sería una excelente iniciativa, que también atrae alumnos, pues varios de ellos están interesados en salir fuera del país por un tiempo.

Aunque en Unicamp aún no estamos demasidado preocupados de cómo atraer estudiantes a CC de pregrado, sí trabajamos en atraer alumnos de magíster y doctorado. Ofrecemos becas de tiempo completo a todos los buenos estudiantes, incluidos extranjeros, y muchas oportunidades para desarrollar investigación en varias áreas interesantes y trabajo multidisciplinario en medicina, biología, agricultura, ingeniería eléctrica, artes y lenguaje. Varios de los candidatos llegan a nuestro programa por esta diversidad.

Por otro lado, iniciativas para atraer estudiantes a CC han sido llevadas a cabo por la Sociedad Brasileña de Computación. Una de ellas son las Olimpiadas Brasileñas de Informática (OBI). Esta es una competencia nacional en programación básica, que en 2009 tuvo 17.000 estudiantes participando, de entre 9 y 19 años. Esta competencia está organizada por la Sociedad y las pruebas son coordinadas en los colegios a lo largo de todo Brasil. Se lleva a cabo en varias etapas. Y los estudiantes que rinden muy bien en todas ellas son invitados a Unicamp a tomar un curso de una semana en programación ofrecido por nuestros profesores. El curso consta de cuatro distintos niveles de dificultad, adaptados a las edades y el nivel de conocimiento de los estudiantes. Al finalizar los alumnos más avanzados realizan un test de programación, y los tres mejores son invitados a participar en las Olimpiadas Internacionales que cada año se celebran en un país diferente. Todo esto es organizado por la Sociedad y financiado por la Fundación privada Carlos Chagas, que paga todos los gastos incluyendo viajes dentro y fuera de Brasil, construcción de pruebas, correción, etc. Este enorme esfuerzo es valioso porque atraemos más alumnos cada año a esta Olimpiada, que comenzó hace aproximadamente 10 años. Varios de los estudiantes que compitieron en los años anteriores están ahora inscritos en carreras de CC. Otra iniciativa anual de la Sociedad es la organización de competencias de robótica durante nuestra conferencia en dicha materia.

Sin embargo, todo esto no es suficiente si se compara por ejemplo con las Olimpiadas Matemáticas, organizadas por la Sociedad Matemática junto con el Gobierno Federal. A estas asisten anualmente millones de niños y jóvenes. Entonces, ¿cuál es el problema con la Ciencia de la Computación? Una de las razones es que no existen suficientes profesores a nivel escolar que puedan motivar a los alumnos a estudiarla o a participar en concursos de programación. Todos los profesores entienden la importancia de las matemáticas, pero muy pocos la importancia de la CC. Por tanto, el mensaje es que tenemos que hacer participar a todos, especialmente a las familias, para atraer jóvenes a la Ciencia de la Computación. Ellos deberían estar concientes de las grandes oportunidades laborales que existen. Pero también de la variedad de trabajos en los que pueden desempeñarse si eligen carreras de nuestra área. La Ciencia de la Computación es mucho más que programación. Sin embargo la imagen asociada es que CC es igual a programación. Y esta imagen debe ser cambiada.BITS

SURVEYS



Modelos de Base de Datos de Grafo

Renzo Angles
Profesor Asistente, Universidad de Talca.Doctor en Ciencias mención Computación, DCC, Universidad de Chile. Ingeniero de Sistemas, Universidad Católica de Santa María, Perú. Líneas de investigación: Bases de Datos, Web Semántica, Ingeniería de Software y Documentación Electrónica.
rangles@utalca.cl

INTRODUCCIÓN

En la comunidad de manejo de la información el término "modelo de datos" ha sido usado de manera amplia y diversa, teniendo de esta manera varios significados. En el sentido más general, un modelo de datos es una colección de herramientas conceptuales usadas para representar entidades del mundo real y las relaciones entre ellas [1]. Desde un punto de vista de base de datos, las herramientas conceptuales que definen un modelo de datos deben especificar la manera de estructurar, restringir, mantener y recuperar los datos. Acorde con estos criterios, un modelo de base de datos consiste de tres componentes: un conjunto de tipos de estructura de datos, un conjunto de operadores o reglas de inferencia, y un conjunto de restricciones de integridad [2].

Desde su introducción, a fines de los sesenta, numerosos modelos de base de datos han sido propuestos, cada cual con sus propios conceptos y principios. Los primeros se enfocaron esencialmente en especificar la estructura de los datos a nivel físico, es decir, de acuerdo al sistema de archivos. Dos modelos representativos son el modelo jerárquico y el de red. Esta perspectiva cambió rotundamente con el modelo relacional, el cual introdujo la noción de nivel de abstracción al separar el nivel físico (implementación) del lógico (modelado). Enriqueciendo el nivel de abstracción, pero desde el punto de vista del usuario, los modelos semánticos permiten representar entidades y sus relaciones de una manera clara y directa. Un ejemplo conocido es el modelo entidad-relación. Basándose en el paradigma orientado a objetos, los modelos orientados a objetos representan

los datos como una colección de objetos organizados en clases y soportando valores complejos como atributos. Posteriormente, los modelos semiestructurados fueron diseñados para modelar datos con una estructura flexible. En estrecha relación se encuentra XML (eXtensible Markup Language): un lenguaje de marcas usado para representar datos semiestructurados para el que se han desarrollado bases de datos. Mayores detalles sobre modelos de base de datos pueden encontrarse en los trabajos de Silberschatz et al. [1] y Navathe [3]. En la actualidad el paradigma relacional es el preferido por la mayoría de los sistemas de administración de base de datos.

MODELOS DE BASE DE DATOS DE GRAFO

Los Modelos de Base de Datos de Grafo (MBDG) se caracterizan porque sus estructuras para esquemas e instancias son modeladas como grafos y la manipulación de datos se basa en operaciones orientadas a grafos. Específicamente, y considerando los elementos de un modelo de base de datos, los MBDG pueden describirse de la siguiente manera:

1) Los datos y/o los esquemas son representados por grafos, o por generalizaciones de ellos (por ejemplo hipergrafos). Un aspecto a considerar es la separación entre los niveles físico y lógico; en este sentido la mayoría de los modelos distinguen el esquema de los datos (instancias).
2) La manipulación de datos es expresada por transformaciones de grafo, o por operaciones orientadas a consultar su estructura (ejemplo adyacencia, caminos, subgrafos, etc.) y propiedades (ejemplo diámetro, centralidad, etc.).
3) La consistencia de los datos se mantiene a través de restricciones de integridad aplicadas a la estructura de grafo. Por ejemplo, consistencia esquema-instancia, identidad e integridad referencial, dependencias funcionales y de inclusión.

La actividad alrededor de los MBDG fue intensa en la primera mitad de los años '90 y desde entonces el tópico casi desapareció. Las razones de esto son, entre otras, que la comunidad de bases de datos se movió hacia los modelos de datos semiestructurados; la aparición de XML capturó la atención de aquellos trabajando en Hipertexto y la Web; los investigadores del área se concentraron en aplicaciones particulares como las bases de datos espaciales o biológicas. Una revisión de los modelos de BD de grafo puede encontrarse en un artículo de ACM Surveys [4]. La Figura 1 refleja el desarrollo del área en términos de los artículos publicados, los cuales describiremos a continuación.

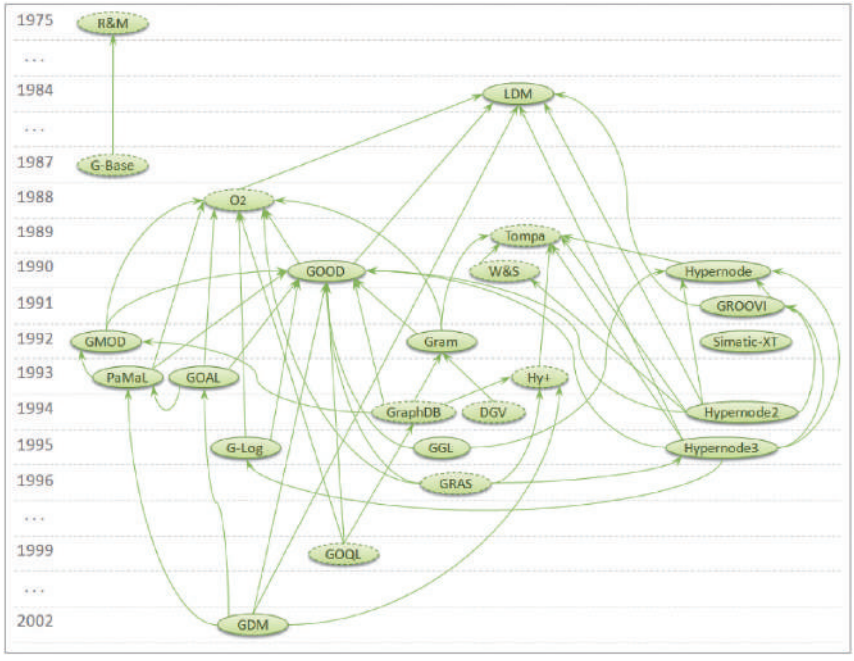


Figura 1. Desarrollo de los Modelos de Base de Datos de Grafo. Las elipses indican modelos y las flechas citaciones. Las elipses punteadas representan modelos relacionados.

En una primera aproximación, y enfocándose en la deficiencias de los sistemas (en su momento) para modelar la semántica de los datos, Roussopoulos y Mylopoulos (R&M) propusieron una red semántica para almacenar conocimiento respecto a una base de datos. Con un objetivo similar, G-Base propuso el empleo de un modelo de grafo para representar estructuras complejas de conocimiento. Un enfoque diferente fue formulado en el Logical Data Model (LDM), donde se intentó generalizar los modelos relacional, jerárquico y de red, a través de un modelo explícito basado en grafos.

A finales de los años '80 encontramos modelos orientados a objetos basados en una estructura de grafo. Por ejemplo, las relaciones entre objetos en O2 se representan a través de un grafo denominado object graph. GOOD es un modelo influyente que buscó aprovechar la orientación a objetos y la naturaleza gráfica de una estructura de grafo. La Figura 2 presenta un ejemplo de este modelo.

Entre los desarrollos subsecuentes, basados en GOOD, tenemos: GMOD, que modela la representación de interfaces de usuario en bases de datos. Gram, un modelo de grafo explícito orientado a modelar datos de Hipertexto; PaMaL, el cual extendió GOOD con una representación explícita de tuplas y conjuntos; GOAL, que introdujo la noción de nodos de asociación; G-Log, el cual propuso un lenguaje declarativo para grafos; y GDM, que incorporó la representación de relaciones simétricas n-arias. La Figura 3 muestra un ejemplo del modelo GMOD.

Por otra parte, encontramos modelos que usan generalizaciones de grafos. El modelo de hipernodo (Hypernode Model) presenta una estructura basada en grafos anidados

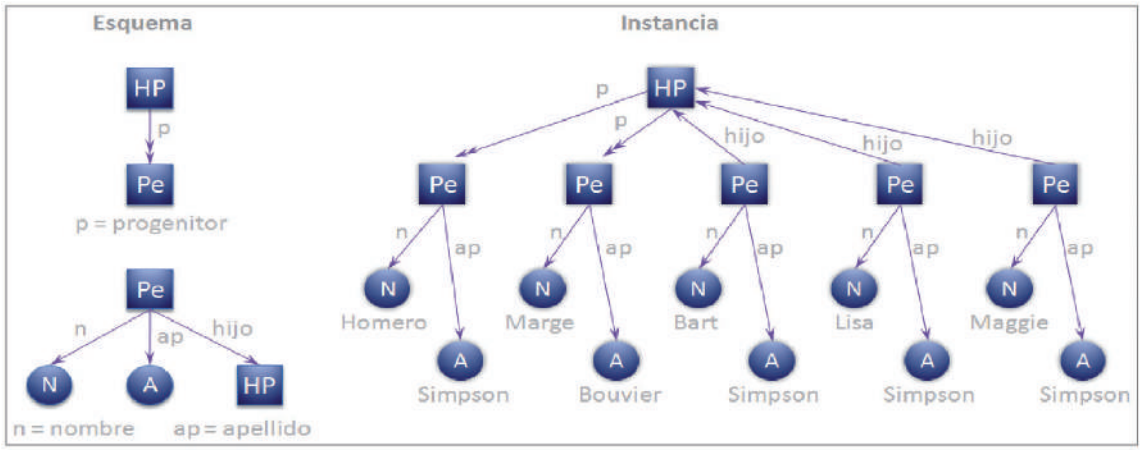


Figura 2. Ejemplo del modelo de base de datos de grafo GOOD. En el esquema: los nodos N y A, denominados imprimibles, representan nombre y apellido respectivamente; los nodos rectangulares representan entidades complejas (Persona) o relaciones (Hijo-Progenitor); una flecha simple indica una relación funcional (mono-valuada) y una doble indica una relación nofuncional (multi-valuada). La instancia consiste en asignar valores a los nodos imprimibles, e instanciar los nodos Pe y HP. En el ejemplo se modelan datos sobre Los Simpson.

(Hypernodes). La misma noción fue usada para modelar redes multiescala (Simatic-XT) y datos biológicos (GGL). GROOVY es un modelo orientado a objetos formalizado a través de hipergrafos (hypergraphs). Esta generalización fue usada en otros contextos como: consulta y visualización (Hy+); representación de esquemas, instancias y acceso a datos (W&S); representación de la estructura y contenido en bases de datos de Hypertexto (Tompa). La Figura 4 muestra un ejemplo del uso de grafos anidados.

Además encontramos otros trabajos relacionados a los MBDG. GraphDB se propuso con la finalidad de modelar y consultar grafos en una base de datos orientada a objetos. Database Graph View (DGV) propone un mecanismo de abstracción para definir y manipular grafos almacenados en bases de datos relacionales, orientadas a objetos, o en sistemas de archivos. El proyecto GRAS usó grafos para modelar información compleja desde proyectos de ingeniería de software. Otra área de desarrollo, reciente e importante, tiene que ver con los modelos de representación de datos en la Web (http://www.w3.org/). Entre ellos podemos mencionar: XML, el modelo de intercambio de datos presentando una estructura de árbol; RDF, el modelo de representación de metadatos basado en una estructura de grafo; y OWL el modelo de representación de ontologías.

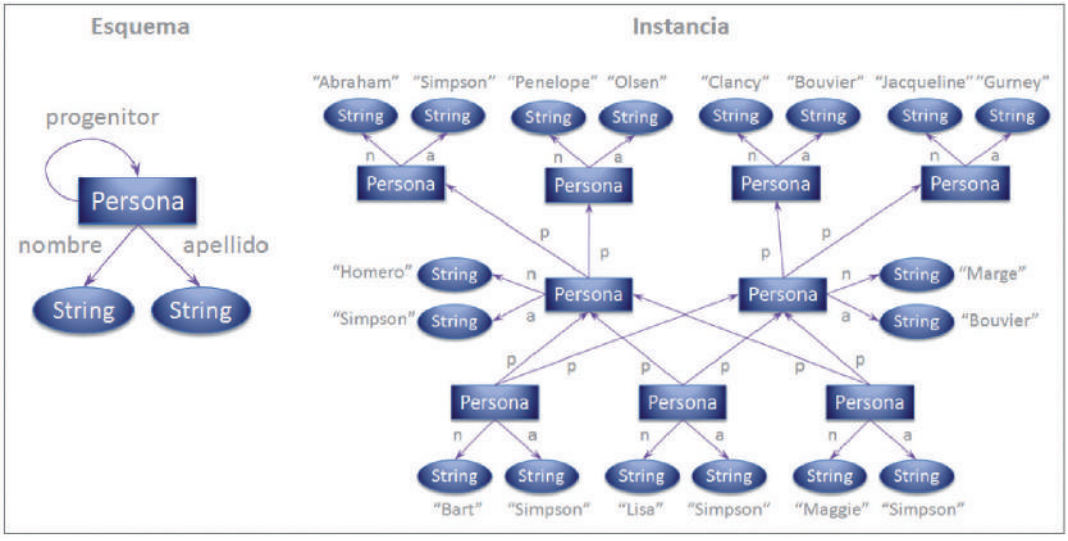


Figura 3. Ejemplo del modelo GMOD. En el esquema: los nodos rectangulares representan objetos complejos (Persona); los nodos elípticos representan tipos de datos primitivos; las aristas representan propiedades (nombre, apellido) y relaciones (progenitor) de los objetos. La instancia se construye creando subgrafos en base al esquema y asignando valores a los nodos elípticos.

MOTIVACIONES

Debido a la importancia filosófica y práctica del modelado conceptual, los modelos de base de datos han llegado a ser herramientas de abstracción esenciales para el desarrollo de sistemas de administración de base de datos (DBMS). La evolución y diversidad de los modelos muestra que hay muchos factores que influencian el desarrollo de los mismos. Algunos de los más importantes son: las características o estructura del dominio a ser modelado; el tipo de herramientas teóricas que interesan a los usuarios esperados; y, por supuesto, las restricciones de hardware y software.

Además cada propuesta de modelo se basa en ciertos principios teóricos fundamentales como es el caso de la teoría de grafos en los MBDG.

Los MBDG son aplicados en áreas donde la información sobre las relaciones entre los datos es tan importante, o más importante que los datos en sí. En este sentido, estos modelos intentan cubrir las limitaciones de los modelos tradicionales con respecto a capturar la estructura de grafo intrínseca en aplicaciones donde la interconectividad o topología de los datos es relevante. Un ejemplo claro son las redes sociales (por ejemplo Facebook), donde la importancia de una persona (dato) se debe principalmente a su red de contactos (relaciones). De hecho, el uso de grafos como una herramienta de modelado trae muchas ventajas para este tipo de datos:

- Modelado de datos más natural. La estructura de grafo es visible para el usuario y permite una manera natural de manejar los datos de las aplicaciones (por ejemplo al modelar una red de contactos). El uso de grafos ofrece la ventaja de mantener toda la información de una entidad en un único nodo, el cual muestra su información relacionada a través de los arcos conectados a éste.

- Consultas propias de grafos. Se tienen operaciones que consultan directamente la estructura y propiedades de un grafo. Por ejemplo, retornar los nodos adyacentes a un nodo, encontrar los caminos entre dos nodos (observe el ejemplo de la Figura 5), encontrar los subgrafos que satisfacen un patrón, etc. Estas estructuras y operaciones de grafos permiten al usuario expresar consultas con un alto nivel de abstracción. En algunos casos no es necesario tener un conocimiento completo de la estructura de los datos para poder expresar consultas. Esto último es particularmente útil al momento de explorar los datos.

- Implementación física ad hoc. Las bases de datos de grafo pueden entregar estructuras y algoritmos especiales para el almacenamiento y consulta de grafos. Considerando la complejidad intrínseca en varios problemas de grafos, se pueden seleccionar estrategias de implementación apropiadas dependiendo de las características de los grafos. Por ejemplo, considere la evaluación de la consulta transitiva presentada en la Figura 6.

APLICACIONES

Los MBDG están orientados a aplicaciones de la vida real donde la interconectividad es una característica clave. Nosotros dividiremos estas áreas de aplicación en aplicaciones clásicas y redes complejas.

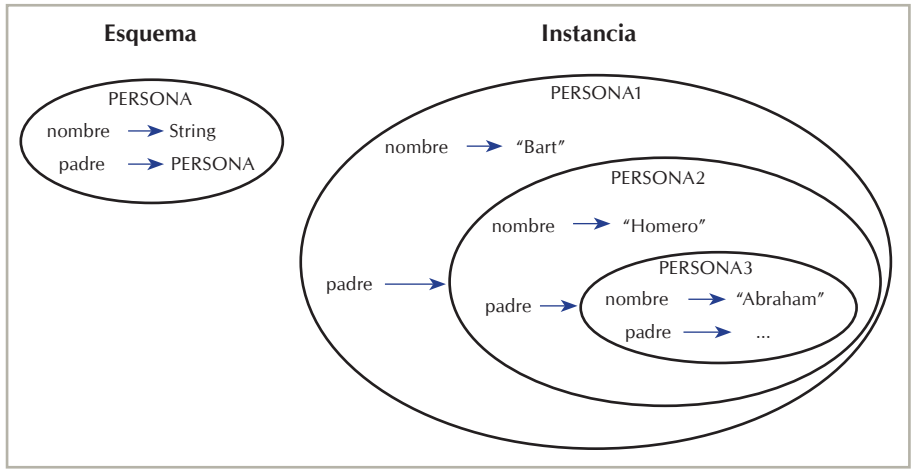


Figura 4. El modelo de Hipernodo (Hypernode model) extiende la definición básica de grafo al permitir que los nodos sean a su vez grafos (hypernodes). En el ejemplo usamos esta característica para representar una estructura anidada de objetos de tipo PERSONA. Cabe resaltar que, aunque no se observa en el ejemplo, el esquema podría definir una relación recursiva (por ejemplo amigo) que podría resultar en una representación cíclica en la instancia.

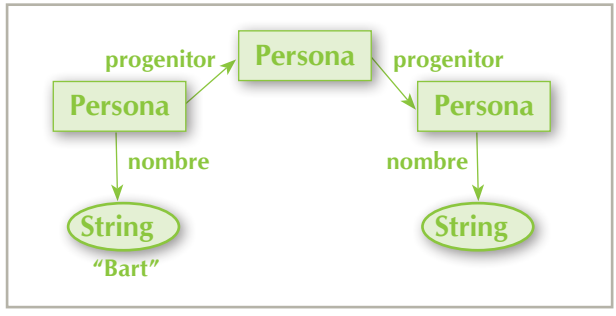


Figura 5. Ejemplo de una consulta de grafo (específicamente de caminos). Considerando el grafo de la Figura 3, la consulta retorna los nombres de los abuelos de "Bart", estos son "Abraham", "Penelope", "Clarcy" y "Jacqueline".

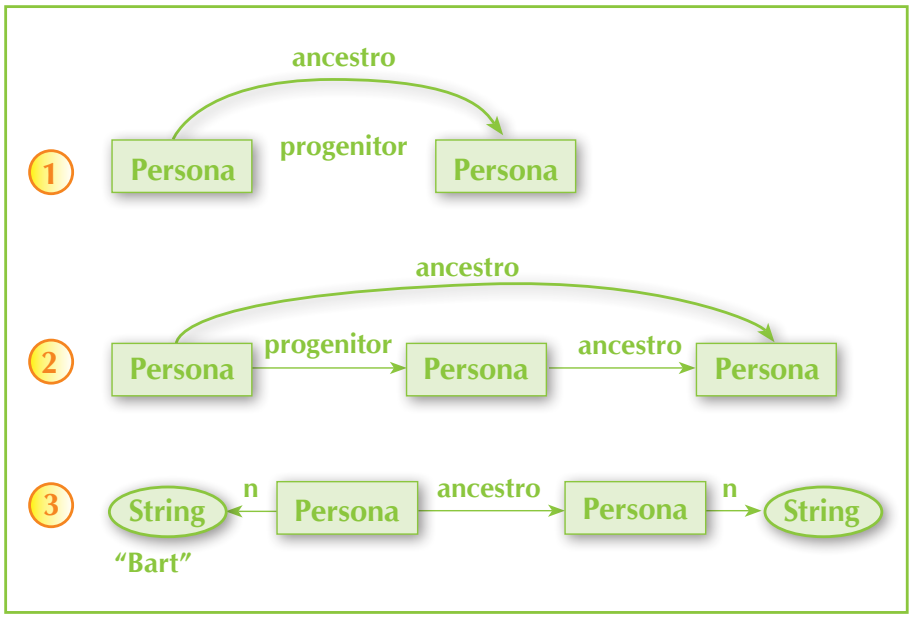


Figura 6. Consulta recursiva en grafos. La consulta retorna los nombres de los ancestros de "Bart" en el grafo de la Figura 3. La consulta define la relación "ancestro" como la clausura transitiva de la relación "progenitor". Esto se expresa a través de tres patrones de grafo: el primer patrón define el caso base para la relación "ancestro"; el segundo patrón define el paso recursivo; y el tercer patrón usa las definiciones anteriores para definir la salida de la consulta.

Aplicaciones Clásicas

El desarrollo de bases de datos de grafo fue motivado por varias aplicaciones clásicas. A continuación describimos algunas.

La generalización de los modelos de base de datos clásicos. Estos modelos fueron criticados por su falta de semántica, la monotonía de las estructuras de datos permitidas, las dificultades del usuario para "observar" las conexiones entre los datos y la dificultad para modelar objetos complejos.

Las aplicaciones donde la complejidad de los datos excede las capacidades del modelo relacional fueron también fuente de motivación. Por ejemplo, los sistemas de gestión de redes de transporte (ejemplo, rutas de trenes y aviones) o las redes en datos espaciales (ejemplo, redes de autopistas y transporte público). Muchas de estas aplicaciones se encuentran en sistemas de información geográfica y bases de datos espaciales.

Las limitaciones en el poder expresivo de los lenguajes de consulta motivaron la búsqueda y definición de modelos que permitieran una mejor representación y consulta de aplicaciones complejas. Otras limitaciones se encuentran en los sistemas de representación del conocimiento, donde se observó la necesidad de técnicas intrínsecas pero flexibles para la representación de éste.

El uso de grafos en el diseño de modelos semánticos y orientados a objetos motivó la idea de definir modelos de datos basados "netamente" en una estructura de grafo. Esto fue acompañado de la necesidad de mejorar la funcionalidad entregada por los modelos orientados a objetos. Por ejemplo, considere aplicaciones como CASE, procesamiento de imágenes y análisis de datos científicos. La aparición de Hipertexto en línea evidenció la necesidad de otros modelos, principalmente semiestructurados. Además la Web creó la necesidad de un modelo de datos más apropiado para la representación e intercambio de información.

Redes Complejas

Diversas áreas han corroborado la aparición de grandes redes de datos con propiedades matemáticas interesantes llamadas Redes Complejas [5].

En las redes sociales [6], los nodos representan personas o grupos y las aristas relaciones o flujos entre los nodos. Algunos ejemplos son las redes de amistad (ejemplo, Facebook), contactos de negocio, patrones de contacto sexual, redes de investigación (ejemplo, colaboración, coautoría), registros de comunicación (ejemplo correo postal, llamadas telefónicas, correo electrónico), redes de computadoras, etc. Hay una actividad creciente en el área de análisis de redes sociales [7], así como en la visualización y técnicas de procesamiento de datos de estas redes.

Las redes de información modelan relaciones que representan flujo de información, por ejemplo, las citaciones entre artículos de investigación, información en la Web [8], redes peer-to-peer, redes de preferencia, etc.

En las redes tecnológicas los aspectos espaciales y geográficos de los datos son dominantes. Algunos ejemplos son Internet (como una red de medios físicos), redes de distribución de servicios básicos (ejemplo, luz, agua, gas, teléfono), rutas y espacio aéreo, redes de entrega postal, etc. Hoy en día, el área de Sistemas de Información Geográfica (GIS) [9] cubre en gran parte esta área.

Las redes biológicas representan información de esta área cuyo volumen ha llegado a ser un interesante problema debido a la necesidad de automatizar el proceso de recolección y análisis de datos. Un buen ejemplo son los datos del genoma [10], con sus problemas de identificación y búsqueda de genes, secuencias metabólicas, estructuras químicas, etc.

Cabe resaltar que los lenguajes de consulta clásicos ofrecen poca ayuda al tratar con los tipos de consulta necesitadas en las áreas mencionadas anteriormente. Por ejemplo, en los GIS se tienen operaciones geométricas

El uso de grafos en el diseño de modelos semánticos y orientados a objetos motivó la idea de definir modelos de datos basados "netamente" en una estructura de grafo. Esto fue acompañado de la necesidad de mejorar la funcionalidad entregada por los modelos orientados a objetos.

(ejemplo, área, intersección, inclusión, etc.), operaciones topológicas (ejemplo, adyacencia, caminos, vecindad, etc.) y operaciones métricas (ejemplo, distancia entre entidades, diámetro de una red, etc.). En las redes genéticas se buscan componentes conectados (ejemplo, interacción entre proteínas), grado de cercanía y vecindad (ejemplo, correlaciones fuertes entre pares). En las redes sociales se mide la distancia entre nodos, el vecindario y coeficiente de agrupamiento de un vértice, tamaño de componentes conectados. En una red de información, como la Web, es natural la necesidad de consultar las conexiones entre los recursos.

RESUMEN

Un modelo de base de datos es una herramienta conceptual que permite definir la estructura de los datos, sus restricciones y la manera de manipularlos (actualización y consulta). Los modelos de base de datos de grafo permiten: modelar naturalmente datos con estructura de grafo, manipular directamente esta estructura, definir restricciones de integridad acorde con ésta e implementar estructuras y algoritmos especiales para almacenar y consultar grafos. Aunque estos modelos perdieron fuerza después de su mayor desarrollo a mediados de los años '80, el incremento de aplicaciones que usan datos con estructura de grafo y las limitaciones de los modelos de base de datos tradicionales (como el modelo relacional) para soportarlas, ha generado el resurgimiento del área. Ejemplos claros de su aplicación son las redes sociales (Facebook), la Web (RDF), información biológica (genoma), etc.BITS

REFERENCIAS

[1] Avi Silberschatz, Henry F. Korth, and S. Sudarshan. Data Models. ACM Computing Surveys, 28(1):105–108, 1996.

[2] E. F. Codd. Data Models in Database Management. In Proceedings of the 1980 Workshop on Data abstraction, Databases and Conceptual Modeling, pages 112–114. ACM Press, 1980.

[3] Shamkant B. Navathe. Evolution of Data Modeling for Databases. Communications of the ACM, 35(9):112–123, 1992.

[4] Renzo Angles and Claudio Gutiérrez. Survey of graph database models. ACM Computing Surveys (CSUR), 40(1):1–39, 2008.

[5] M. E. J. Newman. The Structure and Function of Complex Networks. SIAM Review, 45(2):167–256, 2003.

[6] Robert A. Hanneman. Introduction to Social Network Methods. Technical report, Department of Sociology, University of California, Riverside, 2001.

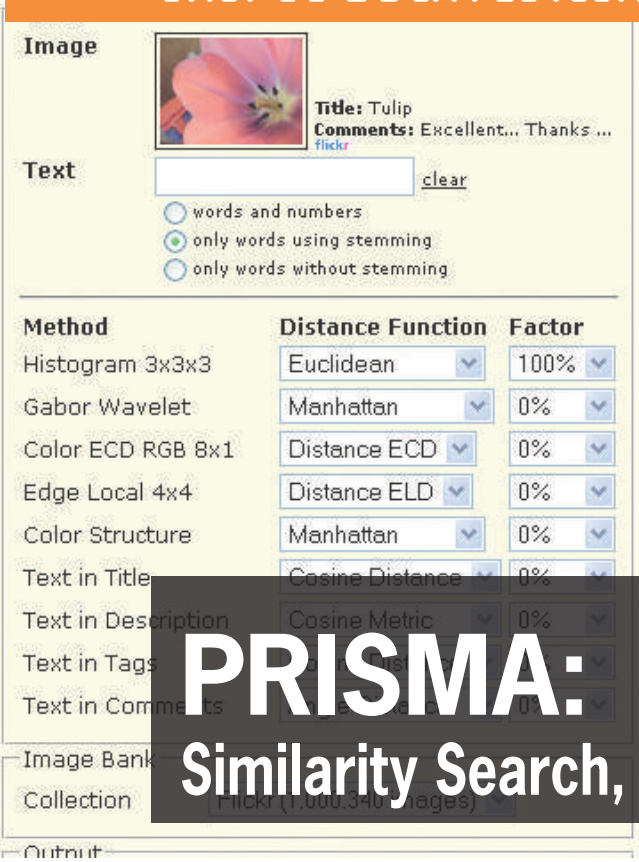
[7] Ulrik Brandes. Network Analysis. Number 3418 in LNCS. Springer-Verlag, 2005.

[8] Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, D. Sivakumar, Andrew Tomkins, and Eli Upfal. The Web as a Graph. In Proceedings of the 19th Symposium on Principles of Database Systems (PODS), pages 1–10. ACM Press, May 2000.

[9] Shashi Shekhar, Mark Coyle, Brajesh Goyal, Duen-Ren Liu, and Shyamsundar Sarkar. Data Models in Geographic Information Systems. Communications of the ACM, 40(4):103–111, 1997.

[10] Mark Graves, Ellen R. Bergeman, and Charles B. Lawrence. Graph Database Systems for Genomics. IEEE Engineering in Medicine and Biology. Special issue on Managing Data for the Human Genome Project, 11(6), 1995.

GRUPOS DE INVESTIGACIÓN



PRISMA:
Similarity Search, and Indexing in Multimedia Archives

PRISMA es un grupo de investigación del Departamento de Ciencias de la Computación (DCC) de la Universidad de Chile, dirigido por el Profesor Asistente Benjamín Bustos. Su objetivo principal es investigar nuevos algoritmos y técnicas para poder realizar búsquedas en grandes colecciones de datos multimedia.

INTEGRANTES DEL GRUPO

Actualmente PRISMA está conformado por su director, doctor Benjamín Bustos; los asistentes de investigación Juan Manuel Barrios, Violeta Chang, José Saavedra e Iván Sipirán; y los estudiantes Víctor Sepúlveda y Diego Díaz.

Benjamín Bustos
Profesor Asistente, DCC, Universidad de Chile. Doctor en Ciencias Naturales, Universität Konstanz, Alemania (2006). Principales líneas de investigación: recuperación e indexamiento de información multimedia y búsqueda por similitud.
bebustos@dcc.uchile.cl

ÁREAS DE INVESTIGACIÓN

Desde el comienzo del nuevo milenio se ha experimentado un fenómeno de "explosión de información", donde el volumen de datos digitales producidos ha aumentado exponencialmente en el tiempo. Esto se ha debido a distintos factores, como por ejemplo la disponibilidad de mejores recursos computacionales, conexiones a Internet de banda ancha y la rápida diseminación de aparatos capaces de capturar información multimedia (cámaras digitales, cámaras de video, teléfonos celulares, etc.).

Actualmente se estima que más de 95 por ciento del material accesible en la Web corresponde a contenidos multimedia. A esto hay que agregar aquellos datos multimedia que se encuentran almacenados en bases de datos corporativas y científicas, archivos personales y bibliotecas digitales. Estas enormes colecciones requieren de algoritmos

y estructuras de datos especializadas para poder realizar búsquedas en forma eficiente y eficaz. Una característica particular de las búsquedas en datos multimedia es que se basan en el concepto de búsqueda por similitud, es decir, el problema es encontrar objetos multimedia que sean parecidos entre sí, ya que la probabilidad de que sean idénticos es muy baja.

Las principales áreas de investigación en las que PRISMA trabaja son:

- Búsqueda por similitud en colecciones de datos multimedia
  - Objetos 3D
  - Imágenes
  - Video
- Métodos de indexamiento
  - Multidimensionales
  - Métricos
  - No-métricos
- Reconocimiento de patrones

PROYECTOS DE INVESTIGACIÓN

Algunos de los proyectos de investigación en los cuales PRISMA trabaja en la actualidad, corresponden principalmente a las tesis del Doctorado en Ciencias mención Computación del DCC de los asistentes de investigación del grupo.

Búsqueda en colecciones de modelos 3D

Los modelos 3D son un tipo importante de dato multimedia que pueden representar información compleja. Problemas de búsqueda por similitud en grandes colecciones de modelos 3D aparecen en muchas aplicaciones prácticas como, por ejemplo, en la industria manufacturera, realidad virtual, aplicaciones médicas, biología molecular y en la industria del entretenimiento. Aspectos interesantes a estudiar con respecto a la búsqueda de modelos 3D son: búsqueda por similitud basada en la geometría global del objeto, búsqueda por similitud parcial (esto es, sólo se conoce una parte del modelo que se desea encontrar), y la búsqueda por similitud funcional (por ejemplo, encontrar si dos modelos 3D pueden ensamblarse para formar un solo objeto).

Búsqueda con medidas de similitud no-métricas

Las aplicaciones para realizar búsquedas por similitud en datos multimedia se han basado, por lo general, en la definición de similitud restringida al caso de distancias métricas. Debido a sus propiedades topológicas, las medidas de similitud métricas pueden ser usadas eficazmente para indexar una base de datos multimedia, la cual es posible consultar de manera eficiente utilizando estructuras de datos conocidas como métodos de acceso métrico. Sin embargo, en los últimos años ha aparecido una demanda por métodos de búsqueda basados en distancias no-métricas, dado que los axiomas métricos son muy restrictivos en ciertas áreas de investigación y sus aplicaciones asociadas.

Por estos motivos es interesante estudiar medidas de similitud no-métricas aplicadas a la búsqueda por similitud en datos multimedia. Dentro de los problemas a resolver en esta área se encuentran el estudio de medidas de similitud no-métricas, cómo pueden ser aplicadas a distintas áreas de investigación, se puede mejorar la eficacia de las búsquedas con estas medidas y cómo es posible realizar búsquedas de manera eficiente.

Detección de copia de videos

La detección de copias de video consiste en recuperar todas las versiones modificadas de un video original dentro de una colección dada. El proceso de detección de copias se puede dividir en dos tareas: descripción del contenido (extraer los descriptores que representan a cada video), y búsqueda por similitud (determinar los conjuntos de descriptores que conforman un buen "calce"). En la búsqueda por similitud, para medir el grado de cercanía entre descriptores, normalmente se utiliza una función de distancia métrica, por ejemplo la distancia euclidiana. Las propiedades métricas representan un compromiso entre eficiencia y efectividad, ya que permiten el uso de los índices conocidos (por ejemplo, el R-Tree). Pero restringen el modelo de similitud que puede ser usado para comparar descriptores. En particular, la función de distancia debe cumplir la desigualdad triangular.

El objetivo de este proyecto es investigar modelos de similitud para secuencias de video que no necesariamente cumplan con las propiedades métricas. En particular, se planea investigar los modelos multimétricos (combinación dinámica de métricas) y no-métricos (funciones que no cumplen las propiedades métricas), con el objetivo de lograr una detección efectiva y eficiente. Los temas involucrados en la detección de copias de video (transformaciones visuales, descriptores globales y locales, dimensión temporal y búsquedas aproximadas) hacen de éste un tópico interesante de investigación y aplicación en distintas áreas.

Búsqueda en imágenes basada en sketchs

Esta investigación se enfoca en una variación de la recuperación de imágenes por

Benjamín Bustos, Violeta Chang, José M. Saavedra, Iván Sipirán, Juan M. Barrios.

contenido conocida como recuperación de imágenes por sketch (Sketch-based Image Retrieval o SBIR), en donde la consulta es un dibujo a mano basado en trazos (un sketch). Las técnicas actuales para SBIR aún mantienen problemas con invariancia a transformaciones (rotaciones, translaciones, etc.), por lo que tienen una tasa baja de efectividad. El objetivo de este proyecto es proponer mejoras tanto en la eficiencia como en la efectividad de los métodos utilizados para realizar SBIR.

Los aspectos que guían esta investigación se centran en: combinación de descriptores, en que se evaluarán descriptores existentes y se propondrán otros que describan mejor un sketch; similitud espacial, en cual se evaluará la relación que tienen tanto las entidades que forman una imagen en la colección de datos como las que forman un sketch de consulta; clustering y clasificación, en que se planea usar estas estrategias de aprendizaje de máquina tanto para reducir el espacio de búsqueda como para realizar retroalimentación en el proceso de consulta; indexabilidad, donde se evaluará el desempeño de los índices métricos en el caso específico de SBIR y se propondrán mejoras considerando el caso particular de los descriptores utilizados. Finalmente, se estudiará el problema de recuperación de imágenes basada en sketchs compuestos por múltiples objetos (MSBIR). Al resolver estos problemas será posible expandir el paradigma de consulta en colecciones de imágenes. Pues los usuarios no requerirán encontrar imágenes que expresen lo que ellos tratan de buscar, lo que puede resultar difícil, sino que podrán bosquejar sus propias consultas.

Búsquedas por similitud usando índices comprimidos

Las técnicas de indexamiento multidimensional enfrentan un gran desafío al ser utilizadas para consultar colecciones de imágenes gigantes: el desempeño de los esquemas de indexamiento multidimensional decae dramáticamente cuanto mayor sea la dimensionalidad de los descriptores utilizados, problema conocido como la "maldición de la dimensionalidad". Las técnicas estándar de filtrado limitan el número de objetos visitados en la colección para responder a una búsqueda por similitud; no obstante, para mejorar la tasa de filtrado en una colección dinámica de imágenes (es decir, que permite agregar, borrar o modificar imágenes) es necesario añadir información adicional.

Por esto resulta interesante estudiar el impacto, tanto en los recursos requeridos como en la precisión de los resultados, cuando en las búsquedas por similitud se usa un índice comprimido con pérdida para los vectores característicos. Por ejemplo, se han propuesto algunas técnicas de compresión de índices para texto con muy buenos resultados de recuperación, que podrían ser extendidos para trabajar con datos bidimensionales (imágenes en este caso). Este enfoque puede ser una solución para hacer que la recuperación por contenido de imágenes sea realmente escalable a grandes repositorios.

COOPERACIÓN INTERNACIONAL

El grupo de investigación PRISMA mantiene lazos de colaboración con investigadores internacionales. Ejemplos de proyectos realizados en conjunto son:

- Búsqueda de Objetos 3D: Dr. Tobias Schreck, Technische Universitaet Darmstadt, Alemania.
- Búsqueda en espacios no-métricos: Dr. Tomas Skopal, Charles University in Prague, República Checa.

INDUSTRIA Y TRANSFERENCIA

PRISMA ha realizado recientemente un exitoso proyecto de cooperación con la empresa chilena Orand, especializada en el desarrollo de software para proyectos de innovación. Este proyecto ha permitido transferir las tecnologías desarrolladas a partir de las investigaciones realizadas por PRISMA, de manera que sus resultados sean aplicados por la industria nacional para mejorar sus procesos productivos.BITS

CONTACTO

Página Web de PRISMA:

http://prisma.dcc.uchile.cl

Contacto: bebustos@dcc.uchile.cl

Universidad de Concepción
Nuevo Programa de Doctorado en Ciencias de la Computación

Por John Atkinson, Loreto Bravo, Leo Ferres, Andrea Rodríguez.

En un reciente estudio realizado por el Consejo Nacional de Innovación se definieron áreas prioritarias o "clusters" de inversión estratégica para el Gobierno de Chile en los próximos años. Dentro de estas prioridades se destaca tecnologías de la información (TI), no sólo como actividad en sí misma sino como un área transversal a otras relevantes tales como la minería, biotecnología y educación, entre otras. Más aún, su relevancia va allende de las fronteras incentivando el desarrollo de la industria del offshoring.

Lo anterior no es nuevo, considerando que es prácticamente imposible pensar en un mundo globalizado sin tecnologías de información, desde su aplicación en las redes sociales en Internet hasta el estudio de fenómenos climáticos, análisis financiero, etc. Más aún, la evolución natural de la ciencia de la Computación en Chile ha generado una mayor necesidad de conocimiento especializado y de alto nivel en problemas complejos del mundo actual.

Como consecuencia de estos requerimientos se está produciendo un acelerado proceso de especialización, comprometido con un enfoque multidisciplinario de solución a problemas no rutinarios tanto en el ámbito científico como tecnológico, lo que impacta naturalmente en la educación superior, en general, y en la formación de posgrado en ciencias, en particular.

Con esta visión y necesidades en mente a nivel regional y nacional, el Departamento de Ingeniería Informática y Ciencias de la Computación, Facultad de Ingeniería de la Universidad de Concepción (UdeC), se propuso el desafío de generar un programa a nivel de Doctorado en Ciencias de la Computación con un grado de especialización tal que satisfaga los desafíos actuales.

EL NUEVO DOCTORADO

El Doctorado en Ciencias de la Computación de la UdeC persigue formar doctores centrado en los intereses específicos del estudiante en función de líneas de especialización desde los inicios del programa. Así, el alumno puede crear su propio programa haciendo investigación en vez de la metodología tradicional basada en cursos estructurados. Si bien las asignaturas están dentro del programa, el doctorando debe tomar sólo aquellos ramos que pertenezcan a su área de interés.

Debido a esta opción por la especialización temprana, el Doctorado de la UdeC pretende que los potenciales estudiantes, al momento de la postulación, tengan una noción clara sobre las áreas de especialización ofrecidas. La ventaja es que esto ayuda a involucrarse tempranamente en investigación, lo que puede llevar a una disminución en los tiempos de duración de los estudios.

LÍNEAS DE INVESTIGACIÓN

El programa de doctorado se concentra en torno a dos grandes áreas de especialización en las cuales se realiza investigación básica y aplicada en la Ciencia de la Computación:

Sistemas de Procesamiento de Información: el área de sistemas de procesamiento de información estudia aspectos formales de ingeniería de software, bases de datos y recuperación de información. Esto incluye aspectos teóricos y de implementación relacionados con la modelación, representación, consulta, manipulación y recuperación de datos o información desde repositorios estructurados, semi-estructurados o no estructurados. Estos repositorios abarcan bases de datos tradicionales, espacio-temporales, multimediales y la Web, entre otros.

Temas específicos abordados por esta área son:

1. Bases de datos y consistencia. Estudia cómo chequear y lidiar con distintos tipos de inconsistencias a nivel de los datos y el esquema, abarcando bases de datos relacionales, XML, espaciales, peer-to-peer y datawarehouse.
2. Desarrollo de sistemas de información. Investiga enfoques dirigidos al diseño de software para la generación automática de código, métodos, técnicas y herramientas que apoyan el desarrollo de aplicaciones Web y, en particular, el desarrollo de sistemas que adaptan automáticamente su navegación y presentación a las características de cada usuario y del contexto de interacción.
3. Sistemas de información especial (espacio-temporal). Investiga la modelación, representación y manipulación de información que tiene una dimensión especial (temporal) bajo tres perspectivas: bases de datos, recuperación de información y razonamiento espacial.

Sistemas Inteligentes: el área de sistemas inteligentes estudia los fundamentos, diseño y desarrollo de sistemas que utilizan modelos y técnicas de inteligencia artificial, para resolver problemas informáticos complejos y aplicaciones en diversas áreas. Esta área incluye: descubrimiento y representación del conocimiento, estudio y desarrollo de agentes autónomos inteligentes, reconocimiento de patrones y minería de datos, técnicas de aprendizaje automático y programación de sistemas evolutivos.

Temas específicos abordados por esta área son:

1. Procesamiento de lenguaje natural. Estudia técnicas y modelos de lingüística computacional e inteligencia artificial para el desarrollo de sistemas que permiten tratar en forma automática el lenguaje escrito o hablado en diferentes medios. La investigación en esta área abarca técnicas y métodos para el análisis automático de lenguaje natural en diferentes formas: diálogos humano-computador, documento de texto.
2. Representación de conocimiento y generación de lenguaje natural. Estudia la generación de oraciones en lenguaje natural a partir de alguna representación semántica de entrada. Los resultados de esta investigación se aplican agregando interactividad lingüística en Non-Player Characters (personajes no manejados por jugadores) en Massively multiplayer online role-playing games (MMORPGs), en traducción automática, en sistemas dialógicos y tutoriales y en la generación de resúmenes en tiempo real de bases de datos numéricas masivas.

3. Aprendizaje Automático. Estudia diferentes teorías, modelos y métodos para la resolución de problemas complejos dinámicos utilizando sistemas que aprendan de los datos y la experiencia. Aplicaciones incluyen minería de datos, agentes inteligentes, robótica, etc.

El impacto de la investigación de estas áreas en la Universidad de Concepción se puede medir en parte por las publicaciones científicas en las revistas relevantes de los tópicos mencionados, y la disponibilidad de financiamiento nacional e internacional tanto del sector gubernamental como del industrial privado. Estos incluyen diversos proyectos y financiamiento proveniente de FONDECYT, CONICYT, IBM y empresas privadas nacionales.BITS

CUERPO ACADÉMICO DEL PROGRAMA

El cuerpo académico del programa cuenta con nueve profesores de jornada completa y cuatro profesores colaboradores:

Profesores	Doctorado	Área de investigación
Julio Aracena	Univ. de Chile	Teoría de computación, matemáticas discretas
John Atkinson	University of Edinburgh	Proc. de lenguaje natural, inteligencia artificial
Loreto Bravo	Carleton University	Bases de datos, sistemas de información
Leo Ferres	Carleton University	Proc. de lenguaje natural, inteligencia artificial
Anahí Gajardo	Univ. de Chile	Teoría de computación, matemáticas discretas
Andrea Rodríguez	University of Maine	Bases de datos, sistemas de información espacial
Gonzalo Rojas	Univ. Politécnica de Valencia	Ingeniería de software-hipermedia adaptiva
Lilian Salinas	Univ. de Chile	Teoría de computación-matemáticas discretas
Javier Vidal	Univ. Politécnica de Madrid	Sistemas Inteligentes-computación gráfica
Colaboradores	**Afiliación**	**Área de investigación**
Leopoldo Bertossi	Carleton University	Bases de datos, lógica computacional
Mónica Caniupán	Univ. of Bío-Bío	Data Werehouse, lógica computacional
Eric Goles	Univ. Adolfo Ibañez	Teoría de la computación, matemáticas discretas
Mauricio Marín	Univ. de Santiago	Sistemas distribuidos y paralelismo

Es importante hacer notar que Leopoldo Bertossi es profesor part-time del Departamento de Ingeniería Informática y Ciencias de la Computación de la Universidad de Concepción, y profesor titular de la School of Computer Science, Carleton University, Ottawa, Canadá. El profesor Bertossi da cursos regularmente cada año en el primer semestre. Asimismo, Mauricio Marín es investigador del centro Yahoo! Research Latinamerica.

Contactos

Páginas web: http://doctorado.inf.udec.cl

Teléfono: 56+(41) 2203564

E-mail: doctorado@inf.udec.cl

Postulaciones en línea: www.udec.cl/graduados

CONFERENCIAS:

Lo Mejor de lo Nuestro

Marcelo Arenas
Profesor Asistente, DCC, Pontificia Universidad Católica de Chile. Doctor en Ciencia de la Computación, University of Toronto, Canadá. Licenciado en Matemáticas, Magíster en Ciencias de la Ingeniería e Ingeniero Civil de Industrias mención Computación, Pontificia Universidad Católica de Chile.
marenas@ing.puc.cl

La importancia de la investigación en Computación es innegable. Y dentro de esta área, la importancia que tienen las conferencias como un medio de difusión de la investigación es también incuestionable. De hecho, muchos de estos eventos internacionales reciben un gran número de artículos de investigación cada año y tienen índices de aceptación menores que algunas de las revistas más prestigiosas de la disciplina.

En el concierto mundial, las conferencias son usadas como un medio para medir el rendimiento tanto de académicos como de grupos de investigación. Y este fenómeno no es ajeno a nuestro país; los grupos de investigación en Computación más importantes de Chile han usado las conferencias como un medio de inserción internacional, permitiendo mostrar que en algunas áreas nuestras universidades producen investigación de primer nivel.

Este año las Jornadas Chilenas de Computación (del 9 al 14 de noviembre de 2009, Universidad de Santiago) tuvieron un evento dedicado a los mejores artículos presentados por académicos de universidades chilenas en las conferencias internacionales más importantes en el área de Computación. La idea detrás fue poder presentar, en un solo día, algunas de las mejores investigaciones de los últimos años.

Para llevar a cabo este evento se me encomendó la tarea de presidir un Comité de Selección de los artículos que se presentaron. Dado que la actividad no contaba con muchas horas para su realización, decidimos concentrar la búsqueda en el período 2006 – 2008, y sólo en las áreas de algoritmos y teoría de la Computación, bases de datos, ingeniería de software, inteligencia artificial y lenguajes de programación. El Comité de Selección quedó constituido por los siguientes académicos:

- **Héctor Allende**, Universidad Técnica Federico Santa María.
- **John Atkinson**, Universidad de Concepción.
- **M. Cecilia Bastarrica**, Universidad de Chile.
- **Narciso Cerpa**, Universidad de Talca.
- **Yadran Eterovic**, Pontificia Universidad Católica de Chile.
- **Claudio Gutiérrez**, Universidad de Chile.
- **Alejandro Hevia**, Universidad de Chile.
- **Gonzalo Navarro**, Universidad de Chile.
- **Andrea Rodríguez**, Universidad de Concepción.
- **Alvaro Soto**, Pontificia Universidad Católica de Chile.

Este Comité se dividió en cuatro grupos, los cuales buscaron artículos en las áreas mencionadas anteriormente. La búsqueda dio como resultado una preselección de 22 trabajos de investigación. Después siguió una fase de discusión que arrojó como resultado 12 artículos, por los cuales sus autores fueron invitados al evento. La siguiente es la lista de investigadores y artículos presentados, junto con las conferencias internacionales donde participaron originalmente:

- **Jorge A. Baier:** Planning with First-Order Temporally Extended Goals Using Heuristic Search (AAAI 2006).
- **Jérémy Barbay:** Succinct Indexes for Strings, Binary Relations and Multi-labeled Trees (SODA 2007).
- **Pablo Barceló:** First-Order and Temporal Logics for Nested Words (LICS 2007).
- **M. Cecilia Bastarrica:** Product Line Architecture for a Family of Meshing tools (ICSR 2006).

  Alexandre Bergel: User-Changeable Visibility: Resolving Unanticipated Name Clashes in Traits (OOPSLA 2007).
- **Loreto Bravo:** Extending Dependencies With Conditions (VLDB 2007).
- **Alejandro Hevia:** Universally Composable Simultaneous Broadcast (SCN 2006).
- **Gonzalo Navarro:** Dynamic Entropy-Compressed Sequences and Full-Text Indexes (CPM 2006).
- **Jorge Pérez:** Semantics and Complexity of SPARQL (ISWC 2006).
- **Guillaume Pothier:** Scalable Omniscient Debugging (OOPSLA 2007).
- **Carlos Valle:** Two Bagging Algorithms with Coupled Learners to Encourage Diversity (IDA 2007).
- **Pablo Zegers:** Exponential Transitions: Telltale Sign of Consistency in Learning Systems (IJCNN 2006).

Para terminar, sólo me cabe agradecer el trabajo y dedicación de los miembros del Comité de Selección. En el proceso de escoger los artículos que iban a ser presentados tratamos de ser lo más objetivos posible. Pero como en todo proceso de este tipo, debemos haber cometido más de alguna falta. La responsabilidad de estos errores debe finalmente ser atribuida al Presidente del Comité de Programa, como diría un buen entrenador de fútbol. BITS

Desde 1975 desarrollando la Ciencia de la Computación en Chile

DOCENCIA • INVESTIGACIÓN • INNOVACIÓN

www.dcc.uchile.cl

Ciencias de la Computación
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
UNIVERSIDAD DE CHILE

REVISTA
BITS de Ciencia
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

UNIVERSIDAD DE CHILE

Ciencias de la Computación
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS UNIVERSIDAD DE CHILE

www.dcc.uchile.cl/revista

dcc@dcc.uchile.cl