REVISTA COLOMBIANA DE COMPUTACIÓN
Volumen 1, número 1
Págs. 7-28

# Diseñemos Todo de Nuevo: Reflexiones sobre la Computación y su Enseñanza

Ricardo Baeza Yates
Dpto. de Ciencias de la Computación
Universidad de Chile
Blanco Encalada 2120
Santiago, Chile
rbaeza@dcc.uchile.cl

El hombre es el único animal que tropieza
dos veces con la misma piedra. *Folklore popular*.
Al parecer, lo único que se aprende de estudiar historia
es que nadie aprende de la historia. *Folklore erudito*.
Todo está altamente interrelacionado.
*Ted Nelson*, el inventor de Xanadu.

**Resumen**

Qué y cómo enseñar son las preguntas fundamentales de nuestro quehacer como profesores. En este artículo presento mi visión personal de nuestra área, un análisis crítico y constructivo y sus implicancias en la educación, incluyendo dos respuestas parciales a las preguntas mencionadas.

**Abstract**

What and how to teach are the fundamental questions in our activities as lecturers. This paper presents my view on these questions related to computer science, and illustrates a critical and constructive analysis and its implications in the education, including two partial answers to these questions.

# 1. Introducción

## Motivación

Comenzando con el nombre de nuestro entorno ya tenemos un problema. ¿Es Ciencia de la Computación y/o Ingeniería en Computación? ¿Es el apellido correcto computador, computación, computacional o informática? ¿Dónde calzan los sistemas de información o son otra área? Yo aún no tengo respuestas claras. Tal vez el problema es intrínseco. Como dice el chiste: la Ciencia del Computador (Computer Science) tiene dos problemas: Computador y Ciencia. ¿Han escuchado alguna vez una ciencia de las lavadoras o de otra máquina? ¿Necesitan las matemáticas o la física decir que son una ciencia? En definitiva es un problema de mayoría de edad, de madurez y, por ende, de inseguridad. En todo caso, es claro que lo que hacemos tiene raíces tanto de la ingeniería como de la ciencia básica. Antes de continuar es necesaria una acotación: muchas de las cosas que expongo son obvias o son de sentido común. Sin embargo, pese a ser obvias, muchas de ellas nadie las dice y por eso están aquí. ¿Serán muy obvias o es después de saberlas que son obvias?

El objetivo final de cualquier software es comunicar cierto conocimiento a la mente de la persona que lo usa y viceversa. El cuello de botella más importante está en las interfaces, en la comunicación final con el usuario, no sólo en su ancho de banda sino en la representación misma de la información (ver Figura 1).

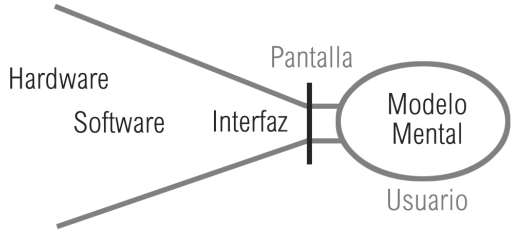


Figura 1: Comunicación de la información.

Cómo resolver este cuello de botella es la motivación principal de este artículo, que puede parecer una mixtura de argumentos, a veces sin relación aparente. Sin embargo, muchas veces olvidamos analizar nuestro universo en forma completa, desde el punto de vista de un observador externo. Para cualquier cosa, tanto el contenido como la forma son importantes en su justa medida y con el balance adecuado. Primero hago un análisis de la relación de la tecnología y la cultura, de la computación en sí misma y del contexto de nuestra profesión. A continuación presento algunas consecuencias de este análisis en la educación y propongo algunas ideas fundamentales con respecto a qué enseñar y cómo enseñarlo. En pocas palabras, mi mensaje es que no olvidemos las relaciones a todo nivel y de todo tipo que existen, que revisemos siempre las hipótesis que hacemos y que hay que rediseñar de verdad y no sólo hacer reingeniería.

## Tecnología y sociedad

Nuestro contexto es altamente tecnológico y por lo tanto es importante entender las relaciones entre nuestra sociedad y la tecnología. La relación entre la tecnología y la cultura es de amor y odio, de éxitos y fracasos, de visionarios y monopolios.

¿Cuánto tiempo necesita la tecnología para permear la sociedad? En muchos casos, pocos años desde el punto de vista de un historiador. Por ejemplo, la imprenta tardó cien años en alcanzar toda Europa. Sin embargo, para una persona de esa época era bastante más que el tiempo promedio de vida. El teléfono o la aviación comercial tardaron más de 30 años para impactar a un porcentaje significativo de la población. El fax fue inventado en el siglo pasado pero sólo en las últimas décadas ha impactado a la sociedad y aún no está presente en la mayoría de las casas. Citando a Norman: Hoy en día escuchamos repetidamente que la velocidad de cambio ha aumentado, que los cambios pasan en "tiempo de Internet", en meses o semanas, no en décadas o años. Falso [29]. La Internet tiene más de 30 años y aún no está en todas las casas ni siquiera en los países desarrollados.

La historia está llena de ejemplos de tecnologías innovadoras o de gran calidad que no tuvieron éxito. Algunos de ellos: Edison inventó el fonógrafo en 1877 y aún así su compañía no tuvo éxito; la primera compañía de automóviles en Estados Unidos nadie la conoce (Duryea); el sistema operativo del Macintosh era muy superior a DOS, pero perdió la batalla comercial; la tecnología Beta de Sony era superior a VHS, etc.

Una de las razones es no entender los verdaderos deseos de los clientes. No siempre la lógica vence a los caprichos del mercado. En el caso de Edison fue la elección de autores para los discos. La gente quería escuchar los cantantes más conocidos. No importaba si habían otros igual de buenos o mejores, es el nombre lo que importa. En el caso de productos, citando a Norman nuevamente: lo que importa es que sea suficientemente bueno para el propósito al que sirve. Más aún, si se lidera el mercado, hasta es permisible usar infraestructura no estándar. Después de todo, si uno tiene la mayoría de los consumidores, lo que uno hace se convierte en el estándar. La competencia no tiene mucha elección aparte de seguirte. Si uno no es el líder, tener infraestructura no estándar es mala idea. Al final, lleva a la extinción [29]. Para software, existen muchos ejemplos de este tipo.

## Productos accidentales, productos dañados

A veces productos malos son vendidos sólo como estrategia de mercado, para dar un primer paso y dar a conocer un nombre. La velocidad es lo esencial, no si el producto funciona o no. Por esta razón, la falta de tiempo es la barrera más importante que impide alcanzar calidad. Muchas veces la calidad tecnológica, un buen diseño, la facilidad de uso (o lo contrario), son igualmente irrelevantes. Citando a Norman: ¿Qué clase de mundo es éste donde no importa que existan productos horribles? El nuestro y no hay otro.

¡Compre! El único computador RISC 100% compatible de 64 bits, sistema operativo Posix compatible y con conectividad total, incluso ATM. La solución para este mundo de sistemas abiertos. Además, gratis el software que Ud. necesita: herramientas de desarrollo orientadas al objeto con interfaz gráfica e inteligente, base de datos transaccional y servidor SQL con soporte de 99 formatos conocidos y por conocer y 64 utilitarios más. ¡Con cuidado! Al igual que en otros productos de este mercado de consumo, lo que la propaganda dice es diferente de la realidad (para confirmar la regla siempre hay excepciones).

La mayoría de las tecnologías más populares e influyentes hoy en computación nunca fueron pensadas para ser usadas como se usan hoy en día ni para dominar el mercado de la manera en que lo hacen. El éxito de MS-DOS/Windows, Unix/Linux, varios lenguajes de programación y World Wide Web demuestran este hecho. No quiero decir con esto que son buenos o malos, sólo que no fueron diseñados para lo que son hoy. Los que conocen la historia de DOS y CP/M sabrán de qué hablo o de cómo prototipos como X-Windows o Mosaic han cambiado la historia. Estos y otros casos son ejemplos

claros de productos y sistemas accidentales que han terminado dominando nuestro mundo y, al mismo tiempo, han debilitado nuestra confianza en poder moldear el futuro tecnológico. De acuerdo con Karl Reed, ésta debería ser una tarea de los profesionales de la computación, los cuales no sólo deberían informar acerca de nuevos avances o problemas sino también dirigir el desarrollo y uso de las tecnologías de la información [30]. Según Reed, esto ha ocurrido por una aversión a planificar. Yo creo que más bien lo que pasa es que cuando planificamos no tenemos éxito por el contexto que describimos a continuación, o no tenemos tiempo para hacerlo.

¿Usted aceptaría un auto en el que tuviera que detener el motor y encenderlo de nuevo para poder usarlo (esto se cuenta como chiste pero en realidad no es gracioso) o un televisor que le diera una descarga eléctrica de vez en cuando? Ciertamente no, pero eso lo aceptamos en software. En marzo de 1999, Microsoft reconoció, en una reunión privada con sus distribuidores, que 5 mil errores de Windows'95 se habían corregido en Windows'98 (¡pero no dijeron cuantos errores habían agregado!). Es decir, que millones de copias de software se vendieron dañadas causando un costo irreparable a los consumidores. En síntesis, necesitamos compañías de software serias y con altos niveles de calidad si queremos enmendar el rumbo. Esto significa cambiar también las políticas de mercado y no conformarse con productos parcialmente terminados.

Lamentablemente, mejores soluciones necesitan tiempo y muchas compañías son reacias a invertir en desarrollo y/o investigación si la tecnología cambia en un año. Esa es la paradoja actual, por cierto entendible, pero dañina. Hoy, en vez de pensar, usamos soluciones ya conocidas que no eran útiles hace 20 años. Es cierto que no es bueno reinventar la rueda, pero tampoco es bueno no tener tiempo para inventar algo nuevo. Llegará el momento donde no tendrá sentido invertir en nuevos avances tecnológicos si no podemos usarlos adecuadamente. Cuando ese momento llegué habrá que volver al pizarrón y pensar. Pensar, algo que la rapidez de esté mundo nos hace practicar poco y olvidar.

## Consecuencias

El uso de la tecnología está determinado en gran medida por accidentes históricos y variables culturales al igual que por la misma tecnología. Los aspectos sociales, culturales y organizacionales de la tecnología son mucho más complicados que los aspectos técnicos. Después que una tecnología se ha establecido y se atrinchera, es muy difícil hacer cambios. Tratemos de recordar esto para el futuro, pues en nuestra área hay muchos ejemplos que exploraremos más adelante, siendo el más obvio el sistema operativo Windows.

Para terminar, recordemos que los avances tecnológicos no sirven de nada si no se dan los avances sociales correspondientes. De hecho, estudios recientes muestran que en realidad la productividad no ha aumentado pese al nivel de informatización logrado en el mundo [6,9].

# 2. Nuestro contexto tecnológico

La tecnología avanza tan rápido que no da tiempo a pensar y diseñar soluciones eficientes con ella. A continuación quiero mostrar como muchas de las soluciones que usamos (y en consecuencia sus diseños) están basadas en suposiciones que ya no son válidas; y como soluciones existentes en el pasado vuelven por sus fueros. Sin embargo, a la larga la carrera tecnológica se vence a si misma. Pero comencemos volviendo atrás hacia algunos acontecimientos.

# Un poco de historia

Tantas veces se han redescubierto las cosas. Pareciera que Windows descubrió las interfaces gráficas, sin conocer la historia de Xerox Parc y luego Apple. Otros creen que la tecnología RISC fue inventada por IBM por su línea de equipos RS-6000, sin saber que fue desarrollada a mediados de los 70. Hagamos un análisis del desarrollo de la computación en estos últimos años. Muchas de las tecnologías han avanzado exponencialmente. Este es el caso de la famosa Ley de Moore que dice que la capacidad de los microprocesadores se dobla cada 18 meses. Esta predicción, realizada en 1965, aún se cumple [15]. Así sucede con la capacidad de un chip de memoria por dólar que ha aumentado 134 millones de veces en los últimos 40 años. Recientemente, un crecimiento similar se puede apreciar en Internet. El número de computadores conectados se dobla cada 15 meses. Esto no puede seguir así pues hoy en día ya más del 20% de los computadores que existen en el mundo están conectados y sino habrían tantos computadores como personas en el año 2010. Por otra parte, el crecimiento de Web es aún más impresionante. Desde 1993, el número de servidores se dobla cada 3 meses, sobrepasando hoy los seis millones. En forma similar, el tramo principal de Internet en Estados Unidos ha aumentado su capacidad en más de 1.000 veces en la década del 80 y posiblemente un valor similar en esta década. Pese a esto, el tráfico en la red crece aún más rápido.

En la Tabla 1 comparamos un computador personal actual con uno típico de hace 16 años. Como observamos, la capacidad de almacenamiento ha aumentado en más de 200 veces y la de procesamiento en al menos 100 veces. Estas diferencias drásticas en desarrollo causan problemas. Por ejemplo, los avances de velocidad en redes (se pronostican Gb por segundo) son difíciles de aprovechar ya que los procesadores no son igual de rápidos. Otras tecnologías no han evolucionado de la misma forma, como en la tasa de transferencia de discos que ha aumentado mucho menos, siendo hoy la entrada/salida uno de los cuellos de botella actuales.

| **Elemento/año** | **1983** | **1999** | **Factor** |
|---|---|---|---|
| Procesador | 5MHz | 500MHz | 100 |
| Memoria | 64Kb | 32Mb | 500 |
| Disco | 20Mb | 4Gb | 200 |

Tabla 1: Desarrollo de los computadores personales.

Por otra parte, los usuarios no hemos siquiera duplicado nuestra capacidad y sin embargo a veces me asombra lo fácil que es acostumbrarnos a algo más grande (como dice una de las acepciones de la Ley de Murphy, no importa de que tamaño sea el disco, siempre está casi lleno). Lo mismo podemos decir del software que tampoco ha tenido un avance espectacular, por no decir que los métodos no han cambiado mucho en los últimos 10 años. Aunque es cierto que ahora muchos de los recursos computacionales son baratos, la solución no es usar el diseño que ya teníamos, sin optimizarlo y pedirle al usuario que se compre un computador dos veces más grande y más rápido.

Las características principales del mundo de la computación actual están dadas en su mayoría por el efecto de Internet. Entre ellas podemos mencionar interactividad, procesamiento e información

distribuida, digitalización y uso de múltiples medios, uso de recursos en forma compartida y sistemas colaborativos, normalización y sistemas abiertos. Es difícil hacer predicciones, muchos erraron en el pasado. Los ejemplos más famosos son del fundador de IBM, Thomas Watson, en 1943: Creo que hay mercado para cinco computadores y del fundador de Digital, Kenneth Olsen, en 1977: No hay razón para que una persona quiera tener un computador en casa. Algunas especulaciones en el corto plazo incluye la masificación de la fibra óptica, el desarrollo de las redes inalámbricas, la convergencia de los PCs y las estaciones de trabajo Unix, el mayor uso de herramientas colaborativas y, por supuesto, la masificación total de los computadores e Internet.

## Sistemas operativos y redes

La mayoría de los supuestos en los fundamentos de los sistemas operativos tradicionales ya no son válidos. En el pasado, los recursos de hardware (CPU, memoria, disco) eran muy caros y se trataba de reducir su uso. Luego, muchas de las soluciones fueron más complicadas de lo necesario, para reducir el costo o el impacto en los recursos compartidos.

Estos supuestos cambiaron en la década de los 80 y junto con el abaratamiento de los costos, también la velocidad de procesadores y memorias aumentó en más de 100 veces. ¿Por qué entonces los sistemas operativos no son al menos 100 veces más rápidos? Para adaptar las soluciones existentes, primero se usaron mejoras en las interfaces (por ejemplo la memoria cache). Pese a ello, un límite máximo dado por la complejidad de la solución misma no podía ser sobrepasado. La solución es simplificar la solución. Este es el paradigma de “rapidez gracias a simplicidad”. Uno de los corolarios de este paradigma ha sido el auge de procesadores RISC en vez de CISC.

De otro lado, se olvida la historia. Windows 1.0 nunca fue vendido, Windows 2.0 fue un fracaso y sólo Windows 3.0 fue un éxito, siendo sólo un buen parche a DOS. Windows NT necesita un mínimo de 16Mb y se sugiere 32Mb. ¿Que pasó con los 64K que necesitaba DOS? ¿Tan barata es la memoria que podemos olvidarnos de ser eficiente? ¿Tan rápidos son los procesadores que podemos olvidarnos de buenas estructuras de datos y algoritmos? Los defensores de Windows NT dirán que es mucho más que DOS, que incluye un sistema de ventanas, conectividad a redes, multiproceso, etc. Bien, pero por ejemplo Linux con X-Windows funciona con 4Mb y mejor si son 8Mb. ¿Por qué entonces Windows NT necesita tantos recursos? Claramente hay un problema de diseño. El auge de la computación móvil puede ayudar a que se mejoren los diseños en este ámbito, pues no podemos darnos el lujo de tener muchos recursos o usar mucha energía (batería).

Un fenómeno similar ha ocurrido en redes. Antes eran caras y lentas. Ahora son baratas y rápidas. La mayoría de las tecnologías actuales han tenido que adaptarse a los cambios, aunque todavía se puede hacer mucho más. Por ejemplo, ATM fue diseñado en los 60s y ahora vuelve a la palestra porque es simple y rápido, alcanzando 155 Mb/s. Pero aún estamos lejos de las velocidades que se pueden alcanzar en fibra óptica, que son de varios Gb/s.

Otro ejemplo es X-Windows, el sistema de ventanas más popular en Unix, que es transparente al protocolo de red usado. Es decir, es un sistema de ventanas distribuido. El protocolo de comunicación usado por X-Windows supone que la red es rápida y que las acciones gráficas en la pantalla son lentas. Hoy en día eso no es cierto porque aunque las redes son rápidas, están congestionadas y son compartidas por muchos usuarios. Por otra parte, la velocidad de las pantallas gráficas también ha aumentado.

# El arte de programar

Programar es quizás el corazón de la Ciencia de la Computación. Es el mundo de los algoritmos y estructuras de datos y de los paradigmas de la programación. A través de su evolución, programar ha sido más un arte que una ciencia o una ingeniería. Por algo la famosa trilogía de Knuth sobre algoritmos y estructuras de datos se titula The Art of Computer Programming [21].

Para muchos programar no es una tarea respetable, para eso están los programadores. Sin embargo, debemos distinguir entre las personas que son capaces de diseñar la solución a un problema y convertirla en un programa, de las personas que sólo pueden traducir una solución a un programa. Un programador real -como diría Yourdon- es el que puede realizar el proceso completo, desde el análisis hasta la implementación.

Programar permite mantener el entrenamiento en la resolución de problemas, ya sean grandes o pequeños. Programar debe ser gratificante. De ninguna manera es denigrante que un ingeniero o lo que sea que pensemos que somos, programe. Al revés, muchas veces nosotros haremos los mejores programas porque somos los que entendemos completamente una solución propia. Otro punto importante es que un buen código no es aquel que no se entiende o es más truculento sino el que es el más claro, eficiente y documentado. Muchos ven también el fanatismo de programar como un sinónimo de ser un hacker. Como cualquier adicción en el mundo, los extremos no son buenos. Tampoco hay que confundir hackers con programadores malvados. Hay hackers buenos y hay hackers malos, y los primeros son imprescindibles.

# ¿Ingeniería de Software?

Hace unos meses, un importante ejecutivo de una gran compañía de computación de Estados Unidos me dijo: podíamos darnos el lujo de hacerlo bien porque teníamos los recursos y queríamos entrar a un mercado nuevo. Por supuesto, a nivel técnico siempre nos gustaría hacerlo bien, pero el mercado dice otra cosa. No hay tiempo, no hay recursos, es ahora o nunca. El resultado son productos mal diseñados y mal probados. Actualmente, la única compañía que podría darse el lujo de hacer las cosas bien en el mercado actual es Microsoft. Pero no pareciera querer hacerlo.

Quizás el mejor ejemplo para comenzar es el famoso problemita del año 2000 o el problema del milenio (aunque en realidad el próximo siglo comienza en el año 2001). Desde cualquier punto de vista, éste es un problema ridículo con un impacto gigantesco. ¿Deberíamos sentir verguenza? Creo que no.

¿Fue un error considerar sólo 2 dígitos en vez de 4? Todos saben que la razón principal fue usar menos memoria, recurso que hace 20 años era mucho más caro que ahora. Yo creo que no fue ni error ni buen diseño. La verdadera razón es que ninguno de los diseñadores pensó que existiría software que permanecería en funcionamiento por más de 20 años. Ni siquiera hoy en día pensamos eso, contagiados con los cambios anuales del hardware. Es cierto que en algunos casos los programas han evolucionado sin cambiar el diseño original, pero no es lo típico. ¿Por qué seguimos usando ese software? Por las malas costumbres en el desarrollo de software, como hemos mencionado.

La computación cambia, pero eso no significa que mejora. Muchas empresas prefieran no cambiar software que sabemos que funciona o que sabemos dónde no funciona, el cual sobrevive a cambios sucesivos de hardware y, por ende, muchas veces se pierde el código fuente original. Otras han

intentado cambiarlo, pero los proyectos han fracasado por no usar las metodologías y/o herramientas adecuadas. Hoy vemos el otro extremo. El uso de recursos es excesivo y el diseño es secundario. Por ejemplo, Windows´98 tiene más del doble de líneas de código que la última versión de Solaris y ocupa mucha más memoria durante su ejecución. El lector puede hacer su propio análisis de cuál sistema operativo está mejor diseñado, sin contar que mientras más líneas de código hay, potencialmente existe un mayor número de errores. No porque la memoria sea hoy en día más barata, debemos abusar de ello.

¿Por qué ocurre esto? Hagamos un paralelo con la ingeniería civil. ¿Se imaginan construir un puente que se cae cinco veces en su período de construcción por errores de diseño? Impensable. Peor aún, se imaginan inaugurarlo para descubrir que hay un error fatal cuando hay 100 personas en él. Imposible. Sin embargo, la técnica de prueba y error es usada por todo el mundo en programación. Otro paralelo es el número de diseñadores. Una casa es diseñada por uno a tres arquitectos. ¿Qué pasaría si fueran decenas? Luego es construida sin realizar cambios mayores al diseño. ¿Cuántas veces el diseño es cambiado por los implementadores? Muchas, porque en parte muchas veces son las mismas personas y el tener dos roles sin separarlos claramente siempre es un problema. Mucho se hablaba de reusabilidad, pero recién ahora con bibliotecas de clases y patrones de diseño (design patterns) esta palabra tiene sentido. En el pasado era difícil aprovechar lo hecho por otras personas por innumerables razones: código no disponible, distintos lenguajes o ambientes, falta de documentación, etc. La modularidad y la independencia de componentes es vital si queremos integrar productos y tecnologías distintas. También se habla de calidad. Junto con reuso y el utilizar herramientas de control adecuadas, es posible que en el futuro podamos hablar de ingeniería de software [17]. Yo, aunque algunos griten al cielo, diría que en la mayoría de los casos es artesanía de software. TeX es el mejor ejemplo, pues inicialmente fue el producto de un excelente artesano, Don Knuth, y hace 10 años que no se encuentra un error en su código (¡y por cada error se paga un monto que crece exponencialmente!). Mientras no cambiemos nuestro modo de pensar y no confiemos en que siempre podemos probar y si hay errores no pasa nada, programar seguirá siendo un arte donde pocos serán maestros y la mayoría serán aprendices. Este cambio debe ser profundo, pues hasta las compañías más grandes de software aún no pueden decir que un producto no tiene ningún error. Los ejemplos de Windows que mostramos a continuación son ilustrativos.

Windows'95 contiene cerca de 15 millones de líneas de código. Usando estimaciones de Caper Jones [19], un código de este tamaño tiene un número potencial de errores de casi 3 millones, lo que sirve para estimar cuantas pruebas hacer. Para reducir este número a cinco mil, esto significa al menos unas 18 iteraciones en las pruebas [23]. Aunque posiblemente las compañías de software debieran realizar más pruebas, esto aumenta costos y retrasa la salida del producto. Lamentablemente la historia muestra que sacar nuevas versiones de forma rápida muchas veces implica un producto exitoso. Esto ocurre porque el consumidor no discrimina conforme a la calidad. Esto es menos cierto para productos críticos, como un servidor Web. Aquí es más importante la calidad y de ahí el dominio del servidor de Apache sobre un sistemas tipo Unix, aunque sea un software de dominio público [28]. Muchas compañías dicen no usar software público porque no tiene soporte. Sin embargo, la mayoría de los productos de PCs, en particular Windows, tampoco tienen soporte. Windows NT tiene alrededor de 25 millones de líneas de código, lo que significa que se deben hacer más pruebas para tener niveles de confiabilidad necesarios. Por otra parte, Windows NT está, supuestamente, certificado en el nivel de seguridad C2 para su uso en Internet. Aún así, un estudio hecho por Shake Communications Pty. Ltd. reveló 104 problemas, algunos de ellos muy serios, que lo hacen vulnerables a hackers [23].

En el caso de software, suposiciones similares a las de los sistemas operativos fueron hechas: recursos caros y escasos. Actualmente, los recursos son baratos y abundantes. Sin embargo, también es malo

abusar de los recursos y escribir software que necesita mucha memoria o mucho espacio disponible en el disco. Esto es válido sólo cuando es realmente necesario, y la mayoría de las veces no lo es. Este es otro efecto colateral de no tener tiempo suficiente para diseñar software y producir para sacar nuevas versiones lo más pronto posible porque así lo exige el mercado. Este abuso de la tecnología tiene un efecto dañino. Por ejemplo, si queremos hacer algo más rápido, la solución más usada es comprar un computador más rápido. Pese a ello, más barato y posiblemente más rápido es usar una mejor solución (mejor software, mejor ajuste de parámetros, mejor configuración de la red, etc.).

# ¿Inteligencia artificial?

Quizá una de las áreas de la computación que más prometió y que menos avances ha logrado es la inteligencia artificial. Ya sea en juegos como el ajedrez o procesamiento de lenguaje natural, los resultados muestran que buenas heurísticas o cajas negras como las redes neuronales tienen efectividad parcial. Sin embargo, aún se está muy lejos del Test de Turing. Me permito usar el ajedrez para exponer mis ideas. En mayo de 1997, Gary Kasparov, el campeón mundial de ajedrez, fue derrotato por Deep Blue de IBM (Big Blue), el campeón de los programas de ajedrez. ¿Ha triunfado la máquina? Analizar este pseudo triunfo de la inteligencia artificial ayuda a poner en el tapete el abuso de términos como sistemas expertos o inteligentes. ¿No es inteligente un buen algoritmo? ¿Es la fuerza bruta inteligente?

A comienzos de los 50 se predijo que en 20 años habría programas que derrotarían al campeón mundial de ajedrez. Se ha necesitado más del doble de tiempo para que eso ocurra. ¿Son los programas de computación entonces inteligentes? No, Deep Blue no piensa como una persona (tampoco piensa, pero digamos que hace algo similar para poderlo comparar). Kasparov sabe qué líneas analizar y estudia en profundidad un número pequeño de movidas. Por otra parte, Deep Blue analiza millones de movimientos y evalúa muchas posiciones, pero lo puede hacer más rápido. La diferencia fundamental es la intuición, la creatividad y la estrategia a largo plazo. Si Deep Blue tuviera la capacidad de evaluar posiciones como lo hace Kasparov, sería invencible. Sin embargo, Deep Blue evalúa una posición con base en heurísticas. Es decir, reglas que funcionan la mayor parte del tiempo, pero otras veces no.

Mientras más complejo sea el juego y mientras el objetivo sea a más largo plazo, más difícil será evaluar una posición dada. Por ejemplo, hace mucho tiempo que el mejor programa de damas es mejor que cualquier humano. ¿Por qué? Porque el número de posiciones en damas es mucho menor y sus reglas son más sencillas, pudiéndose posible evaluar todas las jugadas posibles. Por otra parte, en el juego oriental del Go, donde es necesario ir controlando el tablero poco a poco, sin saber hasta el final si muchas piezas están vivas o no, es más difícil de evaluar porque la estrategia se plantea a largo plazo. En este caso, la intuición y la experiencia son mucho más importantes que la memoria (como en el Bridge) o la capacidad rápida de cálculo (como en las damas).

La primera lectura errada del triunfo de Deep Blue es que puede parecer que el computador ha derrotado al hombre. En realidad, lo que ha pasado es que un grupo de expertos en computación y en ajedrez ha programado un computador de gran capacidad y ha conseguido derrotar al campeón mundial. Es decir, un grupo de personas que ha trabajado durante mucho tiempo, en particular analizando cómo derrotar al campeón, ha logrado más que la inteligencia y memoria de un solo hombre. No me parece tan especial que un programa pueda derrotar a una persona pues la confrontación no es justa. Deep Blue posee una gran cantidad de procesadores, se sabe más de un millón de partidas de memoria y puede evaluar 200 millones de posiciones por segundo. Un experimento interesante sería

comprobar si con menos tiempo por partido, la capacidad de cálculo es menos relevante. ¿Podría Deep Blue derrotar a un grupo de grandes maestros? Lo dudo.

Entre tanto, hay factores ajenos a la inteligencia que afectan la concentración de un jugador de ajedrez. Según algunos ajedrecistas, Kasparov le tuvo mucho respeto a Deep Blue. Otros dicen que tomó muy en serio su papel de defensor de la humanidad y que su derrota sería un hito en la historia. Kasparov es un ser humano, con emociones, que necesita comer, beber y dormir, que siente la presión de saber que no puede influir psicológicamente en el adversario. Un adversario que no comete errores ni se cansa. Si recordamos el pasado, una de las razones de todas las defensas exitosas de su título fue la mayor fortaleza psicológica de Kasparov.

El hombre se derrota a sí mismo todos los días. Kasparov fue derrotado en público. Sólo eso. Cuando un computador pueda leer un libro, entenderlo y explicarlo, ese será un día importante. Por otra parte, Deep Blue es un ejemplo de ingeniería de software, de un buen programa en un mundo con pocos de ellos. Un programa que ha sido mejorado en muchos años, que usa conocimiento de muchas fuentes y que ha tenido tiempo para evolucionar. Si usáramos la tecnología como lo hace Deep Blue, seguramente estaríamos en un mundo mejor.

## Interfaces con sentido común

Por las limitaciones del MacIntosh original que no podía ejecutar dos aplicaciones simultáneamente para que su costo no fuera muy elevado (muy distinto a sus poderosos predecesores: Altos y Lisa), la metáfora de escritorio del MacIntosh no estuvo centrada en los documentos. Por lo tanto, el usuario estaba forzado a seleccionar una aplicación y luego escoger un documento, en vez de seleccionar primero un documento y luego la aplicación a usar en ese documento. Esto, que parece ser lo mismo, supondría una diferencia fundamental en el desarrollo de interfaces. Sólo desde hace algunos años es posible seleccionar un documento y ejecutar una aplicación predefinida o escogerla de un menú. Citando a Bruce Tognazzini, uno de los diseñadores del MacIntosh: Hemos aceptado que la única manera de crear o editar un documento es abrirlo desde el interior de una aplicación o herramienta. Esto es equivalente a introducir una casa entera dentro de un martillo antes de poder colgar un cuadro en una pared o como poner los dientes dentro del cepillo antes de poder lavarlos (del ensayo Nehru Jacket Computers en [33]). A continuación analizamos distintas áreas de la ciencia de la computación, desde lo más básico.

Analicemos las interfaces actuales. La información que almacenamos está basada en una jerarquía de archivos y directorios en la que navegamos de padre a hijo y viceversa. Es decir, en una sola dimensión. Más aún, debemos recordar en qué lugar está y qué nombre le pusimos a cada archivo que creamos (sin incluir las limitaciones de largo, símbolos, o de no poder poner nombres iguales). De otro lado, aunque la pantalla es un espacio bidimensional, la interfaz usa muy poco este hecho y tampoco aprende de cómo la usamos y en qué orden hacemos las cosas. Por ejemplo, podemos mover un archivo a través de toda la pantalla para tirarlo a la basura y justo al final nuestro pulso nos falla. Resultado: dos iconos quedan uno encima del otro. ¡La interfaz podría haber inferido que lo que intentaba hacer era deshacerme del archivo! En mi opinión, parte del éxito de Netscape y el modelo impuesto por HTML es, además de una interfaz muy simple, el tener una estructura de enlaces de sólo un nivel. Nuevos paradigmas de representación visual de conocimiento están ya apareciendo [14].

La tecnología computacional que se usa debería ser transparente para el usuario. De hecho, ¿cuantos usuarios novatos sólo usan un directorio para poner todos los archivos que usan? El usuario no tiene

para que saber que existen directorios o archivos. Además, no todo puede ser clasificado en directorios y archivos. Un archivo debería poder pertenecer a dos o más clasificaciones distintas y estas podrían cambiar también en el tiempo. Como entendemos las cosas depende de nuestro contexto espacial y temporal. Nuestro alrededor no es estático, pero el computador sin necesidad nos fuerza a guardar nuestros documentos de una manera fija en el espacio y en el tiempo.

Pensémoslo bien. El computador debiera -y puede- nombrar y agrupar archivos y recuperarlos usando su contenido o los valores de algún atributo. Por ejemplo, poder decir: mostrar todas las cartas que estaba editando ayer; y obtener las primeras líneas de cada carta, escogiendo de ellas la que necesito. Otra suposición sin base es que necesitamos una interfaz común para todo el mundo. Las personas son distintas, piensan y trabajan de forma distinta. ¿Por qué no tenemos interfaces que se adaptan a cada usuario, que puedan ser personalizadas y que aprendan de la forma y el orden en que hacemos las cosas? Para facilitar la implementación de nuevas interfaces, debemos botar el pasado y reemplazar los sistemas de archivos por datos organizados de manera más flexible y poderosa [4]. Este es nuestro próximo tema.

## Bases de datos

Uno de los mayores problemas de las bases de datos actuales es la diversidad de modelos, aunque el relacional es predominante. Sin embargo, nuevas aplicaciones necesitan datos que no son tan estructurados y rígidos: multimedios, objetos jerárquicos, etc. Aunque existen modelos adecuados para estos tipos de datos, no existen herramientas que permitan integrar bien dos o más modelos. De hecho, los intentos de incorporar estas extensiones en el modelo relacional no han sido demasiado exitosos.

Si abandonamos las hipótesis del pasado, modelos más poderosos y flexibles pueden ser planteados. Un ejemplo son objetos centrados en atributos dinámicos [4]. En este modelo los objetos tienen un número dinámico de atributos, cuyos valores tienen tipo y son también dinámicos. Este modelo puede ser considerado una extensión del modelo orientado a objetos donde no existen clases. Sin embargo, también se define un lenguaje de consulta poderoso que puede manejar conjuntos de objetos que cumplen condiciones arbitrarias en los atributos, incluyendo su no existencia o si tienen valor indefinido. Argumentos a favor de este modelo incluyen su simplicidad, flexibilidad y uniformidad; la eliminación de suposiciones respecto a estructuras de datos y su relación con objetos que contienen información; su facilidad de uso; el permitir múltiples vistas de la misma información a través de consultas; y el hecho de generalizar los sistemas jerárquicos de archivos.

Este modelo debiera simplificar la labor de usuarios, programadores y aplicaciones para trabajar con información. Este modelo es también útil en la Web, donde objetos pueden ser compartidos a través de agregar atributos específicos a cada uso de un objeto. Estos objetos pueden ser manipulados y transferidos en forma abierta usando XML.

## Internet: un nuevo medio de comunicación

La principal comunicación en Internet, el correo electrónico, se basa en intercambio escrito. Esto tiene numerosas desventajas y ventajas, con diferentes consecuencias. Entre las ventajas podemos mencionar el no juzgar a una persona por una primera impresión que es visual, lo que permite muchas veces conocer mejor a esa persona. También muchas personas son menos tímidas al no tener el contacto visual y por ende son más auténticas. Sin embargo, estas mismas ventajas facilitan que

personas ofendan a otras, que se escondan detrás del anonimato o que suplanten o finjan una personalidad, sexo o edad que no tienen. Hay que recordar que ya sea en papel o delante de un computador, uno puede escribir cualquier cosa sin ninguna presión social de educación o diplomacia. Un famoso chiste en Internet muestra un perro diciéndole a otro: nadie sabe que soy un perro en Internet.

Internet es una comunidad nueva, que no posee las características típicas de las comunidades que conocemos. Por ejemplo, relaciones sociales se establecen entre personas que no se conocen físicamente y que probablemente nunca lo harán. El vocabulario de expresiones, gestos, tonos, movimientos de manos que usamos día a día está ausente. La carencia de estos gestos hace más difícil saber si una frase es una broma o un insulto, si una frase es dicha con seguridad o en forma dubitativa, etc. La reputación de cada interlocutor es menos importante porque muchas veces los nombres no son conocidos o no son reales. El anonimato, que muchas veces esconde lo peor de las personas, nunca fue tan fácil. Para poder convivir existen las reglas de etiqueta de Usenet o netiqueta [27]. Aunque muchas de ellas pueden parecer obvias o de sentido común, muchas personas carecen de sentido común (¡tal vez este sentido debiera tener otro nombre!). El castigo más duro en Internet a personas que repetidamente infringen la netequita es el ostracismo. Es decir, la condena al aislamiento absoluto. Pese a la netiqueta hay muchos casos aún a resolver, por ejemplo cuánto debemos explicar del contexto en una conversación para que otros no tengan que preguntar, cuánto debemos contribuir en una discusión sin que otros se aburran, cuándo dos personas debieran continuar una conversación en privado, cuándo podemos gritar (es decir, escribir en mayúscula), cómo diferenciamos entre ser chistoso o vulgar (pese al uso de smiles), entre amistad o avances indebidos, cómo protegernos de ofensas, etc [32]. Otros problemas están en la privacidad de la información y tratar el correo electrónico tan privado como correo normal. También cometer plagio es más fácil, pues copiar algo de la Web, modificarlo un poco y cambiar el autor no cuesta mucho. Pero también la Web ayuda a descubrir más fácilmente estos casos [8,20].

Tampoco podemos asumir una cultura común, ni siquiera para estar de acuerdo que constituye un comportamiento aceptable. Son más de 100 millones de personas de distinta nacionalidad, raza, religión, lengua nativa (aunque inglés es el lenguaje internacional), estudios, etc. Más de 100 países sin fronteras ni leyes comunes. El resultado es libertad, como en ninguna otra comunidad. Todos tenemos la misma voz, formando una democracia perfecta. Al mismo tiempo, no hay nada que impida el caos absoluto y no todos tienen las mismas oportunidades de acceso a ella.

Todo un universo nuevo aparece con el uso comercial de Internet. Por ejemplo, la propaganda indiscriminada es mal vista en Internet. Estos problemas no están resueltos, pero una parte importante de la solución es la existencia del consenso de qué reglas de etiqueta son necesarias. Esto permite que Internet no sea un caos. Además, es indispensable tenerlas para responder a políticos o moralistas que buscan legislar o censurar Internet con reglas mucho más fuertes. Esto mataría esta joven democracia pues la libertad de expresión es el espíritu central del ciberespacio. Es la fuerza que mantiene viva y hace crecer a Internet. Pero al mismo tiempo es su mayor peligro.

Internet también provee nuevas expresiones artísticas. El uso de realidad virtual, animaciones, etc., permiten nuevas formas de diseño. Más allá de museos, exposiciones fotográficas o arte tradicional, aparecen nuevas maneras de crear y expresar. En particular, las páginas personales de Web son el mejor ejemplo. Ya existen millones de ellas y aumentan en forma vertiginosa.

La calidad de ellas no sólo depende del contenido sino también de la diagramación, las imágenes, etc. En este sentido, las mejores páginas son las más vistas, como cuadros famosos en un museo. Pronto tendremos una jerarquía de popularidad de personas basada en números de visitantes en sus páginas personales, que posiblemente reflejen facetas o personalidades que nunca se descubrirían en un contacto cara a cara: un mundo alternativo virtual.

Por otra parte, los enlaces entre páginas, que en cierto modo son votos de confianza o sinónimo de intereses comunes, generan comunidades dentro de la Web. Existen ya varios miles de estas comunidades, las que pueden ser analizadas desde un punto de vista social, ya que evolucionan en el tiempo, creciendo o desapareciendo [22].

# 3. Nuestro contexto profesional

Adicional a los problemas directamente generados por el contexto tecnológico, hay problemas relacionados con el mercado y el contexto profesional. Por ejemplo, el exceso de especialización profesional, la falta de buenos jefes de proyectos de software [37] o la poca interacción entre la teoría y la práctica del desarrollo de software [13]. A continuación profundizamos algunos de ellos.

## Críticas de la industria

Las críticas más recurrentes de la industria son que mucho de lo que se enseña no sirve para nada, que falta aprender más conocimientos prácticos y habilidad para aplicar los conocimientos en el mundo comercial, de modo que la inserción en el mundo real sea más fácil. Que se necesitan ingenieros de software, no científicos [16]. Lo primero que querría decir es que todo eso es cierto, pues depende del punto de vista. Los objetivos comerciales son de corto plazo y los objetivos de la universidad son de largo plazo. Otras críticas específicas incluyen la falta de patentes industriales generadas en las universidades y la falta de innovación empresarial.

Lo que una empresa quiere es una persona joven y con experiencia. La falta de conocimiento práctico es difícil de mejorar en un sistema donde la tecnología cambia tan rápido. Por eso los conceptos son los importantes pues dan adaptabilidad y capacidad de aprendizaje. Es cierto que muchas veces las empresas pierden inversiones realizadas en capacitación, pero es parte de la competencia de mercado. La especialización (por ejemplo, herramientas específicas) y la educación continuada son responsabilidad del empleador, no de la universidad. Pero uno de los problemas principales es que, al invertir en capacitación, el empleador puede perder al empleado al conseguir un trabajo mejor remunerado al estar mejor capacitado. Muchas veces ocurre porque el empleador no valoriza en su justa medida la inversión hecha.

Finalmente, tenemos el aspecto comercial. Creo que este es un problema que va más allá de los ingenieros en computación, es un problema de la interacción de la tecnología con la sociedad. No podemos tener sabelotodos que además son buenos vendedores y tienen capacidad empresarial. Muchas veces estas aptitudes son innatas y no son enseñables (muchas veces siento que estoy tratando de enseñar sentido común y los resultados no son alentadores). Ya hay deficiencias en los programas técnicos dada la cantidad actual de materias distintas que existen en computación y que no pueden ser cubiertas en su totalidad.

# Relación universidad-empresa

La investigación conjunta entre la universidad y la empresa siempre se ha visto empantanada por diversos factores. Entre ellos el lento aparato administrativo universitario y la visión de la empresa, muchas veces justificada, de la inhabilidad de la universidad de cumplir con metas a corto plazo. Hay que desarrollar una infraestructura para investigación aplicada y aumentar la transferencia tecnológica, que es exactamente lo que necesita nuestro país para exportar software y mantener su crecimiento en esta área.
Otra forma de incentivar la investigación aplicada sería crear proyectos de investigación donde sea obligatorio el tener una contraparte industrial, como los proyectos Fondef de CONICYT-Chile, pero no sólo aplicados sino también en investigación básica, pero de montos menores y también grupos de trabajo más pequeños. Esto permitiría abordar problemas específicos y facilitar el apoyo de las empresas, al ser los riesgos menores.

Por otra parte, debemos acercar la universidad a la empresa de un modo útil para ambas partes. Ideas ya mencionadas miles de veces incluyen transferencia tecnológica, estadías cruzadas, etc. Se critica que no hay patentes en la universidad. La misma crítica es válida para las empresas de software chilenas. Primero, toma un tiempo largo. Segundo, no es barato (al menos diez mil US$). Tercero, mientras dura el proceso el resultado no es publicable (que se contrapone al sistema actual de evaluación académica que se basa en publicaciones, aunque esto en Europa está cambiando). Cuarto, las ideas no debieran ser patentables (por ejemplo, un algoritmo).

# Monoposoft vs. Open Source

El llamado movimiento del Open Source (es decir, código fuente gratis) está cada día más fuerte y por ende llama más la atención de los medios. El ejemplo clásico es Linux: ¿Habría su creador imaginado que ahora sería usado por millones de personas? A su vez, Microsoft mantiene una disputa con el gobierno federal estadounidense y su software genera dinero y chistes [23,24]. Lamentablemente muchos de esos chistes nos debieran dar pena en vez de risa. Esconden importantes verdades y generan hidalgos caballeros andantes.

¿Cómo puede ser que no solamente exista software gratis sino que además su código fuente sea público? Esto no tiene sentido en un mercado capitalista, donde sería difícil pedirle a miles de programadores que trabajen sin cobrar. Mi opinión personal es que Open Source existe sólo porqué existe Microsoft. Si tenemos que elegir entre un software barato y uno de Microsoft, por muchas razones, seguramente elegiremos el segundo. Si la alternativa es gratuita, estamos dispuestos a correr el riesgo y a probar ese software. Y, aunque parezca una contradicción, un software gratis puede ser mejor que un software comercial. Si alguna persona encuentra un error y lo informa, en pocos minutos, a través de los grupos de noticias de Internet, podrán recibir una corrección para el problema. Si no, muchos programadores mirarán el código y alguno de ellos encontrará el problema. Este ineficiente mecanismo es a la vez muy efectivo. Otra ventaja es que este proceso es escalable: a medida que el código aumenta de tamaño, más personas pueden involucrarse en el desarrollo. El año pasado un documento interno de Microsoft que habla acerca de los peligros para esta compañía de Open Source se filtró en Internet (ver éste y otros temas relacionados en [31,18,25,26]).

Microsoft es un monopolio de facto. Cada dos o tres años, millones de usuarios deben actualizar su copia de Windows, sin obtener compatibilidad completa con las versiones antiguas y con precios que aumentan con el tiempo: hay que aprovechar el mercado cautivo. Es como tener que cambiarse de

casa frecuentemente y no siempre para mejor. Recientemente, Bill Gates ha publicado sus 12 reglas para el uso eficaz de Internet en las empresas [12], que muestran que está totalmente convertido al mundo del correo electrónico (en 4 años). También ha aprendido las ventajas del software gratis, en particular cuando también permite aniquilar a la competencia: Explorer. Desde el punto de vista económico, el desarrollo de software en Microsoft no es el más efectivo (de hecho, la mayoría de los problemas los encuentran los usuarios, que no siempre pueden obtener ayuda directa para resolverlos), pero sin duda es el más eficiente. Microsoft es tal vez la única compañía que puede parar esta ola de nieve. Hasta podría darse el lujo de desarrollar en 5 años un verdadero sistema operativo y aplicaciones con interfaces mucho mejores, como las que describimos más adelante. Pero esto no va a pasar porque significa ganar menos. Dependiendo del resultado del juicio anti-monopolio y el avance del código público, el próximo milenio será la era de la información o la era de Microsoft.

# 4. Nuestro contexto educacional y algunas propuestas

Me gustaría comenzar citando a Peter Freeman [10]:

*Si comprometemos el núcleo de la computación, corremos el riesgo de perder habilidades básicas de largo plazo. Si fallamos en tomar en cuenta las preocupaciones de los profesionales del área, corremos el riesgo de quedar obsoletos. La clave es alcanzar el balance correcto, pero hay más de una manera de lograrlo.*

Hay que preocuparse tanto de la forma como del contenido. Si se generan buenos profesionales, estos serán agentes de cambio [11]. Ese debiera ser uno de los objetivos principales de la universidad y creo que ha sido también para mi una motivación personal importante para hacer lo que hago.

La mayoría de lo que aprendemos en nuestra vida sirve poco en ella, principalmente conocimiento técnico. Lo importante es la formación asociada a ese aprendizaje, el desarrollo de capacidades lógicas y analíticas, el poder abstraer y conceptualizar y resolver problemas. El objetivo no es conocimiento *per se*, es formativo. Es aprender a aprender constantemente. Creo que este proceso se puede hacer mejor, integrando conocimiento y nuevas herramientas en cursos novedosos, donde el alumno entiende mejor la meta final. Los objetivos principales deben ser flexibilidad, adaptabilidad, enfatizar conceptos y facilitar el aprendizaje continuo.

Todas las ciencias han evolucionado dentro de un contexto real, no por si solas, siendo el origen del cálculo el ejemplo más clásico. Por otra parte, en el pasado hubo personas que conocían gran parte del conocimiento científico humano. Hoy en día esto es muy difícil, forzando al trabajo en grupo y multidisciplinario. Estos dos hechos debieran ayudar a plantear nuevas formas de enseñar.

# Rediseñando la universidad

El primer concepto que deberíamos revisar es el de universidad. La universidad tradicional se basa en tres pilares: la docencia, la investigación y la extensión (relación con la sociedad). Un problema ya antiguo es la utilidad de la investigación básica, de las torres de cristal. Los nuevos tiempos exigen compromisos distintos entre la universidad y la sociedad [7]. Más importante es el impacto de la globalización mundial, las nuevas tecnologías audiovisuales, la Web y bibliotecas digitales, laboratorios virtuales, etc. ¿Debemos aceptar que enseñar es un negocio y que los estudiantes son nuestros clientes? [34]. Posiblemente lo segundo es cierto, pero me cuesta aceptar lo primero. Tsichritzis se basa en los problemas financieros y estructurales de la universidad. En todo el mundo, los presupuestos de la

universidad pública disminuyen progresivamente. Aunque de muchas formas las universidades son monopolios regionales, en el futuro malas universidades pueden tener problemas debido a educación a distancia desde otros países (o hasta educación directa, por ejemplo en Latinoamérica hay muchos MBA españoles). Ya muchos países de habla inglesa comienzan a notar la expansión de universidades estadounidenses o inglesas. Es tiempo de defender nuestros nichos de mercado y una forma es rehusar el contenido que producen [34]. Con respecto a la temática estructural, una universidad produce, programa y distribuye contenido, pero no tiene porqué producir todo el contenido, programarlo o distribuirlo. Puede importar y exportar tanto contenido, como su programación y distribución, aprovechando las nuevas tecnologías ya mencionadas, y posiblemente especializándose en algunos contenidos. En particular el uso de la Web para educación a distancia debe redefinir cual es el verdadero rol del profesor.

# Nuestros estudiantes

¿Que tipo de alumno de computación egresa de la universidad? Y digo egresa, porque un gran porcentaje de ellos nunca se titula. Esto se debe a un fenómeno de optimización local. La demanda por buenos profesionales es tan alta en nuestra área que un alumno es tentado económicamente en el cuarto o quinto año de su carrera con trabajos fuera de la universidad. La resistencia es baja y el alumno comienza a trabajar y estudiar al mismo tiempo. El resultado es una baja en el rendimiento académico, un retraso en terminar sus cursos y, peor aún, un trabajo de título que nunca hacen o nunca terminan. ¿Por qué? Porque a diferencia de tener que cumplir horarios y evaluaciones de cursos, el trabajo de título implica cumplir horarios y metas que uno mismo se define. Y todos sabemos que son metas difíciles de cumplir, más aún si tenemos presiones laborales. Cuando pasa el tiempo, y el alumno ve que necesita su título para progresar laboralmente, es demasiado tarde o mucho más difícil, pues ya ha contraído obligaciones familiares y económicas mayores.

Cada universidad tiene tendencias en su educación que de un modo u otro ocupan nichos de mercado y crean estereotipos. Estos estereotipos están basados en la realidad y muchas veces son ajustados. Por otra parte, están las distintas filosofías educativas. Hay las universidades que enseñan los conceptos, otras que se orientan a productos y máquinas específicas. Es difícil saber qué es mejor, pero en nuestro mundo, productos o máquinas específicas quedan obsoletas en pocos años. Por eso, si alguien me reclama que nuestros alumnos no saben Cobol, o que no han usado sistemas operativos propietarios, o nunca han visto un AS/400, es que aún esta viviendo en el pasado. Un buen dominio de los conceptos permite una mejor adaptabilidad a esta vertiginosa era de cambios.

Por lo anterior, existe entre la universidad y la empresa una brecha tecnológica de varios años (aunque disminuye con el tiempo) y la solución no es que la universidad baje al nivel tecnológico de la empresa. ¿Por qué? Porque creo firmemente que la universidad debe mostrar y hacer el camino con tecnología de avanzada que puede no ser aún necesaria para las empresas, mostrando los caminos futuros y transfiriendo nuevas tecnologías al resto del país. El camino a seguir no puede basarse en la inmediatez de un balance anual. Por ejemplo, en Chile, la universidad fue pionera en el uso de Unix (1984), el uso correo electrónico (1986), el uso de estaciones de trabajo gráficas (1989), Internet (1991), etc. Han sido nuestros alumnos los que han permitido que hoy en día las empresas quieran y puedan usar sistemas abiertos, Internet, orientación a objetos, etc. Por otra parte, hasta mis alumnos de cuarto año trabajan, por lo tanto no parece ser que los profesionales de informática tengan problemas de desempleo debido a que su formación no es la que la industria quiere.

En un área que cambia tan rápido, la única forma de estar al día es entender bien los aspectos conceptuales y aplicarlos a las nuevas herramientas disponibles. Estos conceptos se pueden obtener

a través de cursos de post-título o mediante un estudio personal. Actualmente existen variadas alternativas educativas para actualizar conocimientos en el área computacional o, lo que es más común, debido a la oferta del mercado, cambiar de otra área al mundo de la computación. Hoy en día no nos podemos dar el lujo de no estar al día. Por otro lado, la industria tiene un papel vital. Permite obtener experiencia de largo plazo, de trabajo grupal, con recompensas tangibles y relevantes. Además posee un catalizador abundante: el fracaso. Si aprendemos de él, puede hasta ser un aspecto positivo.

Hay diversos problemas que no permiten la actualización de conocimientos. Uno de ellos es que la empresa prefiere no invertir en capacitación, por el riesgo que tiene de no poder aprovecharla por la alta rotación de los profesionales del área. Hoy, la mayor parte de las veces los conocimientos aprendidos dentro de la empresa misma se pierden cuando el profesional no ve reconocido su nivel y cambia de trabajo a otra empresa que le pagará mejor y que no tendrá que invertir en su capacitación. Alternativamente, muchas veces la persona decide actualizarse por sus propios medios y estudiar al mismo tiempo que trabaja. Muchas veces tiene el apoyo de palabra pero no de hecho de la empresa donde trabaja, pues en la práctica su carga laboral no disminuye.

Finalmente, a cierto nivel empresarial, es útil poseer títulos de postgrado que acreditan su capacidad profesional. Esto es una motivación para volver a la universidad. Lo lamentable es que este afán genera también títulos inexistentes, grados con nombres casi de fantasía y los famosos y ambiguos ‘candidatos a doctor’.

# Diseñando el contenido

Necesitamos nuevos profesionales. Por ejemplo, ingenieros telemáticos, que reúnan el mundo de las telecomunicaciones con el de la informática. Algo así como un híbrido entre un ingeniero electrónico y de computación. Hay que evitar la especialización y destacar aspectos humanos, como ontología del lenguaje, aspectos humanos del desarrollo de software, ética profesional, comunicación oral y escrita, etc. Para diseñar un contenido coherente necesitamos un modelo. En la Figura 2 se muestran las relaciones entre los tres elementos involucrados: personas, procesos y tecnología [16]. También mostramos los mundos que unen estos elementos y que puede ser considerada como una visión dual: mercado, industria y universidad. La ciencia de la computación está centrada en la tecnología, mientras que podríamos decir que el área de los sistemas de información está centrada en los procesos.

Una deficiencia de nuestros currícula (currículos) es que en general se ven las relaciones desde un punto de vista (por ejemplo, cómo afecta la tecnología a las personas) y no el inverso. Por otra parte, no es posible cubrir todas las temáticas en un tiempo limitado y una solución posible es diseñar nuevas carreras. Por ejemplo, un ingeniero de la información que haría el puente entre el proceso de negocios y las personas que lo realizan y la tecnología asociada [5]. Temas importantes que destacan en esta propuesta son, omitiendo los temas tecnológicos y de procesos:

Personas: psicología cognitiva, liderazgo y trabajo en grupo, negociación.

Tecnología-personas: interfaces persona/computador, modelación de conocimiento.

Personas-tecnología: diseño y ergonomía cognitiva.

Procesos-tecnología: gestión de operaciones.

Procesos-personas: administración.

Personas-procesos: recursos humanos y aspectos legales y éticos.

Este modelo, aunque simple, sirve para descubrir las deficiencias y por ende mejorar los programas de estudio existentes. Por ejemplo, la falta de temas como el modelamiento de sistemas [16].

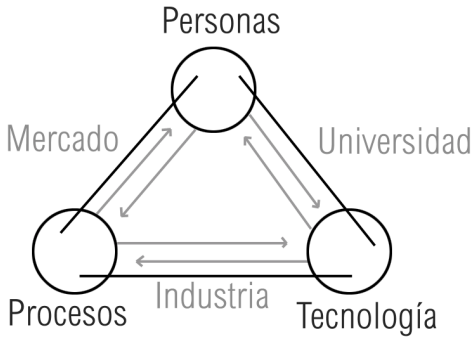


Figura 2: Modelo para diseño del contenido.

# Integrando temas e ideas

En las propuestas de ACM e IEEE para currícula en computación o sistemas de información se hace demasiado énfasis en los contenidos y menos en el cómo enseñar [1,36,2]. Y creo que tan o más importante que el contenido es el cómo enseñar, con base en tres motivaciones principales. La primera es suavizar el paso desde la educación secundaria a la universitaria, donde el alumno pasa de un sistema de educación controlado a uno más libre y con más responsabilidad propia. La segunda es la innovación en la forma de enseñanza, que no ha cambiado en siglos. La tercera es la necesidad de integración del contenido, para aumentar la motivación. Estas tres motivaciones están relacionadas, como veremos a continuación, y nuestra propuesta se centra en la última de ellas, con elementos de las dos primeras.

Con los años uno va perdiendo la capacidad de manejar muchas tareas al mismo tiempo. Hay varias razones que no tienen que ver con la edad. Primero, al aumentar el número de tareas, también aumenta el tiempo que invertimos en cambiar de una tarea a otra (lo que haciendo una analogía con un computador, es el tiempo de cambio de contexto) y segundo, aunque la mayoría de los cambios de contexto son planificados, muchas veces son debidos a interrupciones. Por supuesto, también hay razones externas o estados de ánimo transitorios, que no son controlables. En nuestro desarrollo personal, hay etapas claras en las que aumentamos el número de cambios de contexto y no siempre estamos preparados para ello. Por ejemplo, cuando pasamos del colegio a la universidad y de la universidad al trabajo. El primero es crítico, porque también es una inserción en un ambiente menos controlado y que depende más de las motivaciones y responsabilidades que uno mismo define. Muchos estudiantes universitarios de primer año fracasan porque no se pueden adaptar al nuevo sistema, en especial debido al aumento de cambios de contexto que no pueden controlar. Exámenes, tareas y compromisos siguen uno tras otro sin tener tiempo a planificarlos adecuadamente, pensando en forma repetida si habrá tiempo para hacerlo todo. Un resultado parcial del sobretiempo en cambios de contexto es que actualmente los estudiantes faltan mucho más a clases y tratan de aprobar una materia usando la ley del mínimo esfuerzo. Una acción concreta es disminuir el número de cambios de contexto. Esto se puede lograr enseñando menos materias por semestre pero con mayor profundidad. De manera adicional, no hay que exagerar en la cantidad y tipo de evaluaciones. Existe la creencia errada que los alumnos que más trabajan son los más inteligentes, pero sólo significa que son los más

fuertes física y emocionalmente. Estos cambios permitirían una mejor planificación del tiempo disponible y, es posible, una mayor participación de los estudiantes.

Por otro lado, si en la mayoría de los casos, cuando ingresamos en el mundo laboral debemos trabajar en grupo, ¿por qué la educación sigue siendo principalmente individual? Además de ser casi siempre unidireccional, siendo el profesor el que guía la clase y hace las preguntas. Un argumento importante es que es más fácil evaluar a una persona por su trabajo individual que por su participación en un grupo. Sin embargo, aunque sea más fácil, creo firmemente que debiera motivarse más el trabajo en grupo, donde el profesor juega más bien un rol de comunicador y moderador con base en un contenido ya estudiado previamente por el alumno, permitiendo la comunicación bidireccional entre todas las partes. Esto es lo que se llama educación cooperativa.

Finalmente, tenemos la falta de motivación, por causas variadas y en parte producto de los tiempos actuales. Esta falta es parcialmente producto de que el estudiante no sabe porqué está estudiando una materia dada. Esto ha ocurrido por la especialización del conocimiento convirtiendo cada materia en un compartimento estanco, inclusive dentro del mismo tema. Una solución posible a este problema es aumentar la integración de las materias usando realimentación de temas entre ellas, rescatando las interacciones causales o funcionales que se hayan perdido. Esto se puedo lograr a través de proyectos intercursos, fomentando a su vez el trabajo en grupo y multidisciplinario. Esto puede ser un primer paso hacia una educación cooperativa.

El contenido del curso debe estar basado en las necesidades reales de las carreras de ingeniería. Con base en los contenidos de los cursos de cada carrera se deben determinar qué temas, herramientas y ejemplos típicos es necesario que aprenda un alumno. Esto debiera extenderse a todos los cursos básicos de matemáticas, física, computación y química.

Es necesario un conjunto afiatado de tres o cuatro profesores de alto nivel (posiblemente elegidos a través de un concurso y remunerados en forma especial) que permitan una continuidad en el curso, habiendo previamente discutido y trabajado sobre el material de apoyo. Este último debe incluir herramientas como Web, tutores disponibles vía correo electrónico o talk a horas predeterminadas, software de matemáticas simbólicas (por ejemplo, Maple), etc; incluyendo apoyos tradicionales como ayudantías, laboratorios y apuntes. Además, se debe realizar un estudio de experiencias similares si ellas existen, sus resultados, herramientas disponibles, infraestructura necesaria, etc.

El esquema de clases debería considerar cátedras y ayudantías en forma alternada (por ejemplo mañanas y tardes), incluyendo bastante tiempo de trabajo personal opcional. En forma permanente deberían tratarse dos temas en paralelo (o tres en forma excepcional): más temas sería volver al sistema antiguo, menos temas significaría una carga excesiva en un solo profesor. Por ejemplo podría haber 16 horas de cátedra por mes distribuidas en dos semanas. A esto deberían sumarse 10 horas de ayudantías y unas 4 horas de laboratorio (física, computación). La evaluación se haría en forma continuada y con base en controles cortos (de a lo más una hora).

Para implementar este curso son necesarios varios meses de trabajo intensivo con apoyo de personal calificado para el diseño de los distintos módulos temáticos, la preparación del material, etc. Además, sería conveniente tener primero una sección experimental para evaluar los resultados y compararlos con el sistema actual. Una variante intermedia podría ser una primera mitad del año integrado y luego una separación de materias si se quiere tener un régimen anual. Sin embargo, esto es menos transparente, implica usar métodos mixtos de evaluación y no permite que buenos alumnos puedan avanzar más rápido. Otra forma de experimentar el material del curso (por ejemplo, el comienzo del material) es haciendo una escuela de verano de un mes con alumnos seleccionados.

A continuación se enumeran las ventajas principales en orden aproximado de importancia:

- Integración de materias, facilitando la comprensión y el estudio, al tratarse menos temas al mismo tiempo.
- Eliminación de temas repetidos (por ejemplo conjuntos, espacios vectoriales, algunas herramientas de cálculo, matrices, etc.) que son reemplazados por una visión común.
- Eliminación de materias no necesarias e inclusión de materias necesarias que hoy en día no existen.
- Menos instancias de evaluación y más llevaderas para los alumnos, minimizando el número de interrupciones programadas.
- Posibilidad para combinar actividades de investigación y docencia (por ejemplo estar fuera una semana en un congreso no implicaría pérdida de clases si es planificado con anterioridad).

Las desventajas principales son: se necesita coordinar fuertemente a un grupo de profesores, trabajar con anterioridad en el material del curso, y realizar clases en forma intensiva en períodos cortos. Por esta última razón estimo que es mejor hacerlo en sólo un semestre pues un año completo desgasta mucho al equipo docente.

| |
|---|
| Inducción seguida de derivación de algoritmos para ordenar números. |
| Conjuntos seguidos de estructuras de datos para almacenarlos. |
| Secuencias seguidas de recurrencias. |
| Funciones seguidas de su graficación computacional y el uso de planillas de cálculo. |
| Límites y derivadas seguidas de cinemática simple. |
| Integrales seguidas de trabajo y energía con apoyo de software de matemática simbólica. |
| Vectores seguidos de cinemática en dos y tres dimensiones. |
| Matrices y resolución de ecuaciones seguidos de su resolución numérica vía computador. |

Tabla 2: Ejemplos de integración de materias.

Experiencias similares existen combinando matemáticas discretas y computación, para carreras de computación [35].
Estos ejemplos no incluyen física y no son dirigidos a un plan básico común de dos años para ingeniería. Ejemplos de integración de contenido se presentan en la Tabla 2. Temas que son más generales o previos a temas integrados permiten tener material para paralelizar la cobertura de ellos. Por ejemplo uso del computador (sistema operativo, procesador de texto), herramientas básicas de programación, etc.

# 5. Epílogo

Aunque mis propuestas son preliminares, creo que son un primer paso para diseñar mejores currícula, más completos y coherentes, y enseñarlos de una manera distinta, motivando al estudiante y explicándole claramente por qué está aprendiendo cada tema y las interrelaciones con su entorno. En resumen, integrando todo, volviendo de cierta manera al Renacimiento, al pensamiento enciclopédico e ilustrado. También queda claro que hay que incentivar el pensamiento crítico y enfatizar en los aspectos de diseño.

Este manifiesto personal es a la vez un ensayo sobre los múltiples problemas de nuestra área y un pequeño grito de cordura. Tanto en la vida personal como en la vida profesional, aceptamos tantas

cosas como ciertas, como hipótesis fundamentales, las cuales nunca cuestionamos. Del mismo modo, las ideas expresadas aquí deben ser tomadas sólo como un punto de vista más a ser considerado. Sin embargo, espero que estas líneas apelen a vuestro sentido común, ese sentido tan importante y a la vez tan escaso, y de paso crear un poco de conciencia en los múltiples problemas de nuestra área y de nuestro quehacer cotidiano. Es una crítica constructiva y sin ninguna intención divisiva [13].

## Agradecimientos y notas

Agradezco los comentarios y motivaciones de Omar Alonso, Juan Álvarez, Karin Becker, Tania Bedrax-Waiss, Carlos Castillo, Helena Fernández, Terry Jones, Miguel Nussbaum, Greg Rawlins y Jorge Vidart. Algunos de estos comentarios aún no han sido incorporados en esta versión, que data de 1999.

Alrededor de un 80% del contenido de este artículo es una recopilación coherente de parte de una columna mensual de una a dos páginas publicada en la revista chilena Informática desde 1993 a la fecha, la que me ha dado la posibilidad de divulgar y filosofar sobre muchos temas sin ningún tipo de restricciones ni censura, lo cual agradezco profundamente. Por esta razón, la mayoría de las referencias son posteriores al texto original y, en cierto modo, han corroborado muchas de las preocupaciones e ideas plasmadas en estas líneas.

## Referencias

[1] ACM-IEEE Curricula Recommendations for Computer Science, volumen I, http://www.acm.org/education, 1991.
[2] ACM/AIS/AITP Curriculum for Undegraduate Degree Programs in Information Systems, http://www.acm.org/education, 1997.
[3] Ricardo Baeza-Yates; Terry Jones y Greg Rawlins, A New Data Model: Persistent Attribute-Centred Objects, Reporte Técnico, Univ. de Chile, 1999.
[4] Ricardo Baeza-Yates; Terry Jones y Greg Rawlins, New Approaches to Manage Information: Attribute-Centric Data Systems, Reporte Técnico, Univ. de Chile, 1999.
[5] Ricardo Baeza-Yates y Miguel Nussbaum, The Information Architect: A Missing Link, DCC, Univ. de Chile, Technical Report, 1999. Versión en español en VIII Congreso Iberoamericano de Educación Superior en Computación, Ciudad de México, Septiembre 2000.
[6] Erik Brynjolfsson y Lorin M. Hitt, Beyond The Productivity Paradox, Communications of The ACM, volume 41:8, Agosto 1998, pp. 49-55.
[7] Peter J. Denning, A New Social Contract for Research, Comm. of ACM 40(2), Febrero 1997, pp. 132-134.
[8] Peter J. Denning, Plagiarism in the Web, Communications of ACM 38(12), Diciembre 1995, p. 29.
[9] Sanjeev Dewan y Kenneth L. Kraemer, International Dimensions of the Productivity Paradox, Communications of the ACM 41(8), Agosto 1998, pp. 56-62.
[10] Peter Freeman, Elements of Effective Computation, IEEE Computer, Noviembre, 1997.
[11] David Garlan; David P. Gluch y James E. Tomayko, Agents of Change: Educating Software Engineering Leaders, IEEE Computer 30(11), Noviembre 1997, pp. 59-65.
[12] Bill Gates, Business @ The Speed of Thought, Warner Books, 1999.
[13] Robert L. Glass, Is Criticism of Computing Academe Inevitably Divisive? Comm. of the ACM 42(6), Junio 1999, pp. 11-13.
[14] Ilan Greenberg, Facing Up to New Interfaces, IEEE Computer, Abril 1999, pp. 14-16.

[15] Scott Hamilton, Taking Moore's Law Into the Next Century, IEEE Computer, Enero 1999, pp. 43-48.
[16] IEEE Special Issue, Status of Software Engineering Education and Training, IEEE Software, Nov/Dic 1997.
[17] IEEE Special Issue on the Future of Computing and Software Engineering, IEEE Computer, Enero 1998.
[18] IEEE Special Issue on Linux, IEEE Software, Enero 1999.
[19] Capers Jones, Software Estimating Rules of Thumb, IEEE Computer, Mayo 1996, pp. 117-118.
[20] Ned Kock, A Case of Academic Plagiarism, Communications of the ACM 42(7), Julio 1999, pp. 96-104.
[21] Donald Knuth, The Art of Computer Programming, Volúmenes 1 a 3, Segunda edición, Addison-Wesley, 1998.
[22] R. Kumar, P. Raghavan; S. Rajagopalan y A. Tomkins, Trawling the Web for emerging cyber-communities, Web8, Toronto, Mayo 1998.
[23] Ted Lewis, "Joe Sixpack, Larry Lemming and Ralph Nader", IEEE Computer, Julio 1998, pp. 107-109.
[24] Ted Lewis, What to Do About Microsoft, IEEE Computer, Septiembre 1998, pp. 109-112.
[25] Ted Lewis, The Open Source Acid Test, IEEE Computer, Febrero 1999, pp. 125-128.
[26] Ted Lewis, Asbestos Pajamas: An Open Source Dialogue, IEEE Computer, Abril 1998, pp. 108-112.
[27] J. Martin, Miss Manners looks at netiquette, IEEE Computer 1995, p. 120.
[28] Netcraft Web Server Survey, http://www.netcraft.com/survey, Julio 1999.
[29] Donald A. Norman, The Invisible Computer, MIT Press, 1998.
[30] Karl Reed, Why the CS should help chart the Future of IT. IEEE Computer, Julio 1998, pp. 77-78.
[31] James Sanders, Linux, Open Source, and Software's Future, IEEE Software, Sept./Oct. 1998, pp. 88-91.
[32] Janice C. Sipior y Burke T. Ward, The dark side of employee email, Comm. of the ACM 42(7), Julio 1999, pp. 88-95.
[33] Bruce Tognazzini, Tog on Software Design, Addison Wesley, 1996.
[34] Dennis Tsichritzis, Reengineering the University, Comm. of the ACM 42(6), Junio 1999, pp. 91-100.
[35] A. Tucker; W.J. Bradley y R.D. Cupper, A. Bernet y G. Scragg, Fundamentals of Computing I, McGraw-Hill, 1994.
[36] UNESCO-IFIP, A Modular Curriculum in Computer Science, UNESCO Publications, 1994.
[37] Mark Weisz, Dilbert University, IEEE Software, Sept./Oct. 1998, pp. 18-22.