Arbor

Modelos informáticos para entender la evolución (Mecanismos de evolución adaptativa facilitada)

Juan Cedano

Arbor CLXXII, 677 (Mayo), 101-125 pp.

«La generación espontánea de la vida en la tierra habría tenido tantas probabilidades de darse como el ensamblaje de un avión 747 por un tornado a su paso por un depósito de chatarra». Fred Hoyle

El modelo darwinista es un modelo sencillo que pretende explicar la manera en la que los seres vivos han evolucionado. La simplicidad de sus planteamientos ha facilitado la difusión de estas ideas. Pero esa misma simplicidad hace que difícilmente pueda explicar la complejidad de los seres vivos. Tampoco explica la enorme velocidad de evolución de las especies, que se deduce de la aparente ausencia de intermediarios en el registro fósil. Para soslayar estos problemas, diversos autores han sugerido la presencia en la célula de mecanismos adaptativos y no preadaptativos, como sugiere el modelo darwinista.

La arquitectura de los sistemas informáticos, así como los lenguajes de programación, han tendido, a medida que se han ido haciendo mas complejos, a adoptar la filosofía de los servicios prestados o programación orientada a objeto. Otras herramientas como el debugger han facilitado tremendamente la tarea de los programadores para depurar código y localizar y corregir errores de programación. La adopción de esta filosofía de trabajo ha posibilitado la aparición de sistemas informáticos complejos, robustos y de fácil actualización. Los principios que han posibilitado el aumento en complejidad, su robustez y han facilitado la evolución de lo sistemas informáticos, también podrían estar presentes en la célula. A estos mecanismos los podríamos llamar de «adaptación facilitada», y permitirían: a) introducir azar «controladamente» en los genes, b) acotar los conjuntos de genes sobre los que este mecanismo actúa, c) «validar» si los cambios introducidos han sido o no adaptativos, d) facilitar la fijación de dichos cambios en el genoma y e) facilitar la propagación horizontal, mediante virus, de genes adaptivos.

Introducción

En los organismos vivos encontramos diferentes niveles de organización: molecula, célula, tejido, órgano, aparato, individuo, grupo, población, comunidad, etc. Aunque en este texto tan sólo se hará referencia al nivel celular, se ha de tener en cuenta que los mismos principios que se van a definir para la célula se puede extrapolar al resto de niveles de organización. Mecanismos análogos a los que se van a describir para la célula ya se han descrito con detalle para el nivel organizativo individuo. Esta información, por desgracia, es de difícil acceso, aunque sí existe algún libro que puede proporcionar un poco información sobre el tema (Herrera$_a$, 1999; Herrera$_b$, 1999).

El planteamiento general de este capítulo es utilizar los sistemas informáticos como fuente de inspiración, para tratar de comprender los mecanismos moleculares y celulares que podrían estar implicados en los procesos evolutivos. Se pretende hacer incapié sobre ciertos indicios que apuntan que podrían existir en las células ciertos mecanismos que dirigen y facilitan la adaptación de los seres vivos. En el texto se hace un recorrido sobre algunos aspectos del funcionamiento de los seres vivos, estableciendo numerosos paralelismos entre la organización de la información en los seres vivos y los ordenadores.

En los últimos años hemos asistido a la aparición, expansión y evolución de los sistemas informáticos. A lo largo de esta evolución, tanto máquinas como sistemas operativos y aplicaciones se han ido haciendo cada vez más complejos. Gestionar de forma efectiva esta creciente complejidad ha obligado a los ingenieros de sistemas a estructurar y jerarquizar la información y los recursos. De esta manera, se impidió que este aumento en complejidad condujese indefectiblemente a convertir nuestras computadoras en sistemas cada vez más inestables. Los seres vivos, al igual que los sistemas informáticos, podríamos ser definidos como sistemas complejos de información dinámica. Es muy probable que durante el largo proceso evolutivo que condujo a la aparición de los organismos pluricelulares, algunos de los mecanismos que han permitido a las computadoras evolucionar como sistemas estables y coherentes, también se han hecho patentes en la organización interna de los seres vivos. Estos mecanismos no son más que una respuesta a la complejidad creciente y permiten acotar el esfuerzo adaptativo de los seres vivos comprometiendo lo menos posible el funcionamiento global del individuo.

Cuando se habla de evolución, normalmente aparece la visión oficial del tema, que es básicamente Darwinista. En esta visión se habla de mutación al azar generadora de varianza preadaptativa guiada por la «to-

dopoderosa» selección natural. Este modelo de evolución difícilmente explica la falta de intermediarios en el registro fósil, y la enorme velocidad de aparición de las nuevas especies. En depósitos aluviales como los del lago Turkana se observa claramente esta discontinuidad en el registro fósil. Los fósiles de los moluscos del lago están profusamente representados y se estratifican ordenadamente pero sin intermediarios, a pesar de que según Williamson no se aprecian cambios significativos en el entorno del lago. ¿Dónde están estos supuestos eslabones perdidos, que el registro fósil se empeña en ocultarnos?

Existen multitud de indicios que nos permiten sospechar que el verdadero funcionamiento de los sistemas biológicos no se asemeja demasiado a los mecanismos propuestos por Darwin. En este texto no se pretende realizar una descripción pormenorizada de toda la bibliografía que se aparta de los modelos Darwinistas, tan sólo se mostrarán los mínimos ejemplos que nos permitan hilvanar un hilo conductor más o menos inteligible. De todas formas, esta tarea ya la han realizado algunos autores, que se han dedicado a recopilar exhaustivamente documentación discrepante con el modelo Darwinista. De entre ellos destacaría el libro de Rémy Chauvin, «Darwinismo El Fin de un Mito». En este libro se recogen multitud de referencias que permiten al lector generar una visión crítica sobre la visión Darwinista de la evolución y sus implicaciones. Pero tanto en éste como en otros libros críticos con la obra de Darwin, falta la propuesta de un modelo alternativo. La crítica sólo tiene sentido si es constructiva, y no se puede dejar a la comunidad científica sin un modelo.

Es cierto que el azar actúa sobre los sistemas vivos. Por ejemplo, durante la replicación del DNA. Estos errores introducen cambios en la secuencia de DNA que podrían considerarse azarosos. Gran parte de estas modificaciones serán mutaciones silenciosas[1], o no se traducirán en un cambio fenotípico lo suficientemente acusado como para dar lugar a un cambio significativo en «fitness» para el individuo que la hereda. Pero, por lo general, las mutaciones al azar no producirán una mejora, sino más bien todo lo contrario. Estas mutaciones al azar y la varianza que de ella se deriva, desde el punto de vista informático, no serían más que ruido de fondo, simplemente errores del sistema.

Para un programador, la idea de que por azar una aplicación pueda mejorar en cualquier aspecto, es a todas luces absurda. Los sistemas vivos son sistemas de alta complejidad que contienen una gran cantidad de información que puede ser usada dependiendo de los requerimientos del sistema. En el fondo, entre una célula viva y un ordenador se pueden establecer muchos paralelismos que nos ayudan a comprender los meca-

nismos que hacen posible la evolución. La información de los sistemas complejos, aunque puede estructurarse de muchas maneras, ha de cumplir una serie de requisitos que contribuyan a dar estabilidad y coherencia interna a los mismos.

Una pequeña visión histórica puede ayudarnos a aclarar conceptos. En los albores de la programación, los ordenadores tenían que ser programados en código máquina. Esto implicaba, por parte del programador, conocer la arquitectura interna de la máquina con que se trabajaba, conocer las llamadas a las diferentes funciones disponibles en el microprocesador, así como los registros en los que se debían depositar los datos y aquellos registros en los que se recogen los resultados. Esto hacía de la tarea de programar un trabajo tedioso y poco intuitivo. Pero en cierta manera, y debido al elevado precio de los ordenadores y la memoria, era la forma óptima de trabajar. Con el tiempo, y gracias al descenso en el precio de los ordenadores y sus componentes, fue extendiéndose el uso de código fuente en lenguajes de programación cada vez de más alto nivel, más cercano al lenguaje del usuario, ya fuera éste lenguaje compilado o interpretado. Los sistemas operativos fueron tomando progresivamente más importancia, asistiendo al usuario y evitando en gran medida los conflictos a la hora de gestionar los recursos de la máquina. La evolución de los lenguajes de programación les llevó a incorporar progresivamente una mejor gestión de los errores, tanto de las aplicaciones como del sistema. Al correr del tiempo, se pasó de la programación lineal a la programación estructurada. La programación estructurada era más intuitiva que la programación lineal, hacía que el código fuese más inteligible, facilitaba la reutilización del código y permitía descomponer las tareas más complejas en multitud de funciones básicas con entidad propia. Estas funciones básicas son accesibles y pueden ser llamadas desde otras funciones para dar lugar a nuevas tareas complejas.

La forma en la que el genoma de las bacterias se organiza seguiría, básicamente, un modelo de programación estructurada.

Programación estructurada en la organización de los sistemas vivos

Para un usuario informático el funcionamiento interno de su ordenador es como una caja negra. El usuario no es «consciente» de la gran cantidad de operaciones que internamente realiza el ordenador para que, por ejemplo, el texto que se teclea usando un procesador de textos aparezca finalmente por pantalla. Si todo funciona correctamente, como

usuarios, sólo tendremos «conciencia» de las respuestas para las que nuestro ordenador ha sido programado. Pero eso no siempre es así, en la pantalla también pueden aflorar mensajes que nos advierten de algún error interno en el sistema. Si se crea un programa en el que se intente dividir cualquier número por cero, el sistema no podrá hacerlo. En lugar de devolver el resultado al usuario, el sistema advertirá que para esa operación no tiene respuesta apropiada. Nuestro sistema informático presenta una cierta tolerancia a los errores, es capaz de detectar algunos errores para los que está programado como la división por cero. Pero incluso en aquellos errores para los que no está específicamente programado, el sistema es capaz de advertirnos que se está produciendo una operación errónea en dicho sistema y que la aplicación ha generado dicho error o simplemente no responde. La información que se obtiene del sistema puede ser muy importante a la hora de depurar los errores de las aplicaciones instaladas en la máquina. Ahora bien, si la máquina en la que se ejecuta dicha aplicación errónea pertenece a un programador, muy probablemente dicho error provocará la ejecución automática de una herramienta que los programadores utilizan habitualmente para depurar el código que generan. Esta herramienta de depuración de código suele formar parte del compilador que el programador utiliza para implementar y compilar sus aplicaciones. El control que el programador puede ejercer sobre los procesos que realiza la aplicación se hace aún más patente cuando se activa la utilidad del «*debugger*» mientras se ejecuta la aplicación en el compilador. El «*debugger*» permite controlar la ejecución de la aplicación paso por paso. Y además, permite cambiar el contenido de las variables en cualquier momento de la ejecución de la aplicación sin necesidad de volver a recompilarla[2].

Esta forma de funcionar nos parece de lo más «natural» en un ordenador. Entendemos que se trata de una forma lógica de actuar para localizar errores y abordar los problemas, facilitando en gran manera la tarea de los programadores para generar nuevas aplicaciones. Ahora bien, *¿qué consecuencia tendría sobre el proceso evolutivo de la vida si en los seres vivos existieran mecanismos similares a los descritos en los ordenadores?*

La respuesta a esta pregunta es bastante obvia. Para los programadores estas herramientas suponen una reducción considerable en los tiempos de desarrollo de una nueva aplicación. En un ser vivo, representaría que su velocidad de adaptación podría ser mucho más alta que la que esperaríamos por puro azar. Pero, *¿existen mecanismos similares entre los seres vivos?*

Veamos un posible ejemplo biológico de lo que se intenta decir. En 1991 Barry B. Hall publicó un artículo en el que se analizaba la tasa de

reversión de un doble mutante trp-. Cada una de las dos mutaciones por separado, trpA o trpB, en sus respectivos genes, impide que la célula pueda sintetizar triptófano. Para calcular la tasa de reversión de cada una de las dos mutaciones por separado se realizaron sendos cultivos en placa. En colonias en fase estacionaria, se observó la aparición de papilas[3] capaces de crecer a pesar de que el triptófano del medio ya se había agotado. La tasa de reversión de la cepa que presentaba las dos mutaciones se calculó de la misma manera. Si realmente la adaptación de la bacteria a condiciones de deficiencia de triptófano fuese debida a mutaciones al azar, la tasa de reversión del doble mutante sería fácilmente calculable, simplemente habría que multiplicar la tasa de reversión de la mutación trpA por la de trpB. Sin embargo, el resultado del experimento fue tremendamente sorprendente. La tasa de reversión era 100 millones de veces mayor de lo esperado. Es obvio que, en las bacterias de dicho experimento se estaba produciendo un mecanismo de evolución guiada y facilitada. Además, Hall secuenció los revertientes y observó un hecho curioso, en los mutantes del gen trpA se apreciaban tres tasas de crecimiento muy diferentes. Un subtipo de colonia crecía igual que las células silvestres, pero en los otros dos subtipos se apreciaba una tasa de crecimiento más baja. Estos dos últimos subtipos, aunque producían un enzima activo, éste era distinto al del enzima silvestre.

Este experimento se aleja bastante de un modelo Darwinista. Pero, ¿podría ajustarse a los supuestos que antes hemos definido? Existen algunos puntos sobre los que habría que hacer un especial énfasis. En primer lugar se ha de considerar que la bacteria se encuentra en un estado de deprivación (*starvation*) por falta de un aminoácido esencial. Dicho de otra manera, la supervivencia de esta bacteria se encontraría seriamente comprometida por falta de triptófano. Esto podría disparar mecanismos de respuesta similares a los que se producen en los ordenadores. Pero, si bien el sistema operativo de un ordenador puede identificar qué aplicación no responde, otra manera más burda y simple de saber qué ha fallado es mirar qué aplicación provoca el error al ser ejecutada. De esta manera descartamos todas las otras aplicaciones, que aun estando instaladas en nuestro disco duro, no estaban en ejecución en el momento de producirse el error de sistema.

Si se aplica este sencillo principio al experimento de Hall, se aprecia que, entre los genes que están siendo expresados en el momento de hacerse patente el «conflicto biológico», se hallan los genes de síntesis de aminoácidos esenciales. Se aprecia que al haberse agotado el triptófano del medio, la bacteria ha entrado en fase estacionaria, con la consiguiente inhibición de gran parte de la cascada de genes que rige la división y

el ciclo celular. En este momento el azar sí que podría jugar a favor de la célula, introduciendo a pequeñas dosis mutaciones en el RNA mensajero de los genes que se están expresando hasta encontrar una respuesta adaptativa. Esta respuesta adaptativa, en el caso que nos ocupa, vendría en forma de RNA mensajero revertido que produciría proteína nativa, reconstruyéndose así la ruta metabólica. En el momento en que dicha pseudo-reversión se produjera, el sistema tendría que fijar el RNA producido y recombinarlo con los genes de origen para así poder de fijar la adaptación conseguida. En una bacteria la posibilidad de realizar un banco de pruebas, utilizando el RNA, sin necesidad de pasar por una ronda de replicación, recuerda tremendamente a alguna de las utilidades típicas de un «*debugger*».

Estos sencillos mecanismos moleculares permitirían producir varianza justo en los caracteres que la célula necesita cambiar para adaptarse, reduciendo así los «daños colaterales» de la mutación al azar y reduciendo el papel determinante que la selección natural ejercería, según el modelo Darwinista. La célula, al revertir esta mutación «resuelve» su «conflicto biológico» y vuelve a la normalidad. Introducir cambios en los genes de control del metabolismo del triptófano dejará de ser un objetivo para la célula, y ésta saldrá de lo que se podría llamar compilación en «*debugger*».

El «conflicto» se «resuelve» aunque esa respuesta sea el 30% del crecimiento de una cepa silvestre, como sucede con el gen trpA. No se trata de una respuesta de todo o nada, los sistemas biológicos se ajustan más a las características de los sistemas difusos.

Los mecanismos propuestos, pueden explicar parcialmente una tasa de reversión 100 millones de veces superior a la esperada por azar, e implica una regulación fina de dichos procesos.

Existen algunos indicios que apunta a que podrían haber funciones en las RNA polimerasas bacterianas que todavía desconocemos. Eso explicaría porqué la evolución ha construido una proteína multimérica compleja, cuyas funciones pueden ser aparentemente emuladas por RNA polimerasas monoméricas de origen vírico. De todas formas, no es el interés de este texto describir detenidamente todas las moléculas implicadas en los mecanismos de «*debugger* celular» en los diferentes organismo, ya que una descripción pormenorizada no haría más que introducir más confusión en el modelo. Por lo que nos centraremos, para tener una visión global del «*debugger* en la célula» en los mecanismos propios de las células eucariotas de animales «superiores». En el resto de organismos, aunque el nombre con el que se ha bautizado a las proteínas y las cascadas de activación del «*debugger*» puedan variar, las funciones que el «*debugger* celular» desempeña son equivales.

El «debugger *celular*» en las células eucariotas animales

La herramienta «*debugger*», aunque está mucho más refinada en los lenguajes de programación orientada a objeto, también la encontramos en los compiladores de lenguajes estructurados. Antes de entrar en los mecanismos moleculares concretos en los que pueden estar implicado el «*debugger* celular», hay que remarcar que la principal diferencia entre la evolución en las células y en los ordenadores es que la célula no cuenta ni con programadores, ni ingenieros de sistemas; en este caso la función innovadora y organizadora que el programador ejerce sobre el código ha de ser suplida por otros mecanismos moleculares que permitan introducir modificaciones «controladas» en el genoma.

Qué elementos o mecanimos han de integrar el «debugger celular»

• *Ha de incluir algún mecanismo que permita introducir mutaciones en el código genético.* La aparición de mutaciones está bien documentada en determinadas condiciones fisiológicas, como por ejemplo en los tumores. La proteína mejor estudiada es la p53[4] expresada en altas concentraciones en células tumorales y a la que se le atribuye una función antitumoral. El estudio de esta proteína puede ayudar a entender cómo funcionaría el «*debugger* celular». Cuando observamos la frecuencia de mutación de la proteína p53, encontrada en multitud de tumores, se aprecia claramente que la tasa de mutación a lo largo de la proteína es tremendamente variable. Hay regiones de la proteína en las que prácticamente no se han encontrado mutaciones y otras regiones en las que las mutaciones son tremendamente variables (Van Oijen *et al.*, 2000). Esta variabilidad en la frecuencia de aparición de mutaciones puede estar relacionada con el concepto de **abstracción** explicado más adelante en el texto. Bajo la perspectiva de la «adaptación facilitada», nos podría estar indicando que los intrones no son simples secuencias intercaladas en los exones, sino que pueden incluir información para la RNA polimerasa que, cuando funciona en condiciones de «conflicto activo», puede variar su tasa de fidelidad de traducción dependiendo de si la secuencia que se traduce ha sido marcada en los intrones como variable o como conservada. Ahora bien, hablando de forma estricta, el «*debugger* celular» permitiría introducir mutaciones controladas en el RNA mensajero producido, pero fijar dichas mutaciones en el DNA implicaría mecanismos de recombinación. La recombinación es uno de los mecanismos de más alta eficiencia a la hora de introducir o revertir mutaciones (Ellis *et al*, 2001; Beetham

et al., 1999; Liu *et al.*, 2001). Dicha recombinación estaría haciendo en realidad las veces del programador, ya que modificaría el código fuente del programa (DNA) tras probar una modificación con el «*debugger*» que solventa con éxito un error del programa.

• *El «debugger» ha de permitir relacionar el código con su proceso en ejecución.* En un ordenador esta relación es directa ya que sólo se está ejecutando un proceso. En el caso del «*debugger* celular», para que el acoplamiento entre código mutado (RNAm mutado) y proceso (proteína) se produzca, es necesario que la vida media del RNAm sea larga. Una larga vida del RNAm permitiría producir la suficiente proteína para testar si los cambios introducidos en el RNAm son o no adaptativos. Paralelamente se debería producir una baja tasa de traducción de la RNA polimerasa, ya que si se están produciendo a la vez muchos RNAs mensajeros mutados es difícil determinar a cuál de ellos atribuir una posible mutación adaptativa. Ahora bien, *¿existe algún indicio de que estos mecanismos acoplados existan en la célula?*

Hasta hace poco tiempo, estos mecanismos tan sólo eran parte de un modelo teórico. En estos momentos existen algunos indicios que sugieren su existencia física en la célula. La roscovitina es un potente inhibidor reversible de varias quinasas dependientes de ciclina, proteínas implicadas en el control del ciclo celular. Cuando se administra esta droga se observa un fuerte incremento en la cantidad de proteína p53 que se acumula en el núcleo. Este efecto está íntimamente ligado a la inhibición de la transcripción de la RNA polimerasa II (Ljungman, 2001). Este dato parece indicar que realmente existe un mecanismo que permite bajar las tasas de transcripción de la RNA polimerasa II. Pero sigue faltando un elemento en el conjunto, ha de producirse un aumento de la vida media del RNAm acoplado al resto de procesos propios del «*debugger* celular». Tradicionalmente se ha asumido que la poliadenilación del RNA mensajero tan sólo se producía en el citoplasma y, dado que la transcripción se da en el núcleo, entre ambos procesos no parece existir una conexión directa. Ahora bien, hace poco se ha descubierto una nueva poli(A) polimerasa, sobreexpresada en núcleo de células cancerosas, a la que se ha llamado neo-PAP (*Poly(A) polymerase*) (Topalian *et al.*, 2001). Para que la proteína neo-PAP se ajuste totalmente al modelo su actividad debería ser modulada de igual manera que la de la RNA polimerasa. Pues bien: neo-PAP contiene un motivo conservado de reconocimiento por ciclina y múltiples lugares de forsforilación ciclina dependientes (Topalian *et al.*, 2001). En otras palabras, y a falta de experimentos concretos que lo demuestren de forma más fehaciente, esto podría querer decir que la poliadenilación nuclear viene acompañada por una baja tasa de transcripción del DNA.

• *El «debugger» permite depurar el código de las aplicaciones, pero no se puede usar para modificar el propio sistema operativo.* En el «*debugger* celular» debería suceder lo mismo: sobre el RNAm de determinadas estructuras celulares, que podríamos considerar el núcleo del sistema, no se debería aplicar ninguna tasa de reducción de la fidelidad de transcripción. El ribosoma podría ser considerado una parte fundamental del núcleo del sistema. La estructura de un ribosoma es tremendamente compleja, y éste está compuesto por numerosas proteínas y por RNA. El RNA ribosomal, de hecho, representa la mayor partel de RNA que se produce en el núcleo. Si la transcripción de los ribosomas estuviera ligada a la RNA polimerasa II, al entrar la célula en «conflicto activo» el gran número de mutaciones que se podrían acumular en el RNA ribosómico generaría rápidamente ribosomas inactivos que colapsarían la maquinaria celular. Por este motivo, tanto la transcripción del RNA ribosómico (r-RNA) como la transcripción de RNA de transferencia (t-RNA) estarían ligados a una RNA polimerasa distinta de la II. Concretamente, se sabe que la RNA polimerasa III transcribe el r-RNA en el nucleolo y que la RNA polimerasa I transcribe el t-RNA.

• *Fijación de los cambios adaptativos en el código fuente.* Este mecanismo, como ya hemos mencionado anteriormente, sería realizado por el programador en un sistema informático, pero en la célula ha de ser el propio «*debugger* celular» el que se encargue de hacerlo de forma automática. Esto implicaría que la cascada que se pone en marcha en el «*debugger*» también ha de facilitar la recombinación. Parece que hay datos que así lo indican: efectivamente, la proteína p53 puede activar la transcripción de la topoisomerasa I (Albor, 1998), necesaria para este proceso.

• *Truncado de árboles de exploración.* Esta idea está íntimamente ligada a los mecanismos del «*debugger* celular». Consiste en descartar aquellos caminos pretendidamente adaptativos que pudieran conducir a callejones sin salida. Cuando una célula en «conflicto activo» pone en marcha su «*debugger* celular», pero en lugar de resolver su «conflicto» el camino que ha tomado empeora su situación, desde el punto de vista de la «adaptación facilitada» sería conveniente destruir la célula, y también las posibles modificaciones que sobre el DNA esta célula hubiera hecho. Este mecanismo es la apoptosis. El bloqueo de la RNA polimerasa II y la inducción de la expresión de la p53, si se llevan al extremo, conducen a la apoptosis celular (Ljungman, 2001; Lane, 2001).

• *Otros mecanismos.* La activación del «*debugger* celular» también debería facilitar la adopción de otros mecanismos adaptativos como el *trans-splicing*, la incorporación de genes no propios, ya sean víricos, bacterianos... etc.

Programación orientada a objeto en la organización de los sistemas vivos complejos

Volviendo a los símiles informáticos, el gran salto en complejidad de los sistemas informáticos forzó la adopción de la programación orientada a objeto. Los seres vivos tampoco detuvieron su evolución tras la aparición de las bacterias. La programación orientada a objeto aporta una serie de ventajas y, de haber sido adoptada por los seres vivos, les proporcionaría un procedimiento de gestión interna de la información tremendamente útil que permitiría una evolución rápida de los sistemas vivos.

¿Qué ventajas aporta la programación orientada a objeto?

1. Da lugar a sistemas más robustos y con menos errores.
2. Aumenta la productividad permitiendo construir módulos reutilizables.
3. Es prácticamente imprescindible cuando nos enfrentamos a grandes sistemas complejos.
4. La utilización de la herencia[5] contribuye de forma decisiva a crear un sistema más fiable, fácil de ampliar y menos costoso de mantener.
5. Genera menos código y en módulos más comprensibles.
6. La encapsulación[4] de los datos y las funciones ayudan a diseñar sistemas más estables, impidiendo que la modificación de una estructura, función o procedimiento pueda afectar a otros procedimientos.

¿Qué inconvenientes presenta?

1. La ejecución de estas aplicaciones resulta más lenta.
2. Requiere desarrollar bibliotecas de clases.

La implantación generalizada de la programación orientada a objeto fue propiciada por el considerable aumento de la capacidad de los ordenadores. Pero, aunque en un principio representó un considerable incremento del número de horas que los programadores y analistas empleaban en definir las estructuras de datos, las funciones y los procedimientos a ellas asociados, con el tiempo y la utilización de librerías de objetos predefinidos y de propiedades como la herencia, –usada para de-

finir nuevas clases y objetos–, se acabó consiguiendo una evolución rápida de aplicaciones con una elevada complejidad, minimizando además los errores. Este tipo de programación es mucho más intuitiva, pues entre otras cosas, facilita gestionar eventos, muy importante, en la programación de entornos visuales amigables a los que estamos ya tan habituados.

Si la programación estructurada se interesa primero por los procedimientos y después por los datos, el diseño orientado a objetos se interesa en primer lugar por los datos, a los que se asocian posteriormente los procedimientos (Ceballos, 1997). Esto es, ahora la idea no es tanto qué ha de hacer esta función, sino más bien, ¿sobre qué trata este programa?[6]. Se construye, no alrededor de los procedimientos, sino alrededor de los datos, ya que estos suelen ser más estables.

Elementos de la programación orientada a objeto

Estas características las encontraríamos de forma más clara en las células eucariotas. Algunas de las características fundamentales son: abstracción, encapsulamiento y herencia, y sus mecanismos básicos son: objetos, mensajes, métodos, clases y subclases.

A continuación se definirán estos conceptos, aunque en realidad se trata de conceptos muy intuitivos.

Abstracción

La abstracción consiste en la generalización conceptual de los atributos y propiedades de un determinado conjunto de objetos. Precisamente la clave de la programación orientada a objetos está en abstraer los métodos y los datos comunes a un conjunto de objetos y almacenarlos en una clase. Desde este punto de vista la introducción o eliminación de un objeto en una determinada aplicación supondrá un trabajo mínimo o nulo (Ceballos, 1997).

Este concepto desde el punto de vista biológico está íntimamente ligado al de herencia, que detallamos a continuación.

La abstracción permite trabajar con el DNA independientemente de la función biológica que éste realice. Esta propiedad podría explicar la existencia de intrones en la secuencia de genes. Supongamos una célula con un «conflicto de asimilación». En el medio hay una fuente de carbono que no es asimilable por la célula. Finalmente la célula, siguiendo la es-

trategia de cambio mínimos, consigue introducir cambios de especificidad de sustrato en unos enzimas y esto le permite a la célula digerir la nueva fuente de carbono, resolviendo así su conflicto. Este nuevo gen será colocado junto a la batería de genes hidrolíticos. Ahora bien, de la secuencia del propio gen, independientemente de la función que éste realice, se puede obtener la información de qué partes de la proteína han variado y cuáles no lo han hecho con respecto al enzima original. Esta información puede ser muy interesante para futuros intentos de adaptación, ya que nos permitiría ir clasificando las partes de la proteína que conviene conservar y aquellas que son mutables. Los intrones podrían marcar la separación entre regiones fijas y variables. Esta estrategia es muy interesante, ya que la acumulación progresiva de esta información podría acelerar tremendamente la velocidad de adaptación mediante el mecanismo de *trans-splicing*, que permite recombinar regiones de diferentes proteínas. La presencia de intrones también debe de permitir la compactación de código gracias al *splicing* alternativo.

Encapsulación

Esta práctica se refiere a incluir dentro de un objeto todo lo que este objeto necesita. De esta forma el objeto actúa como una caja negra. También limita el acceso a los datos y funciones incluidas en un objeto a los demás objetos de la aplicación, siempre y cuando se tenga en cuenta la recomendable práctica de declarar funciones y datos como privados. La encapsulación y el objeto están íntimamente ligados, y se basan en la filosofía de servicios prestados y caja negra.

No existen los modelos puros. De la misma manera que con lenguajes estructurados no orientados a objeto se puede programar siguiendo los criterios de la programación orientada a objeto, en las bacterias también se puede encontrar ciertas características propias de la programación orientada a objeto. Un ejemplo de una cierta encapsulación en los genes eubacterianos lo encontraríamos en los diversos factores σ que pueden integrase en el pentámero que constituye la RNA polimerasa y que confieren a ésta especificidad respecto a determinados tipos de promotores (Rojo, 1999). Este cambio de especificidad por parte de la RNA polimerasa permite a *Bacillus* pasar a expresar genes de esporulación tan sólo cambiando el factor s que se integra en el pentámero.

En organismos pluricelulares la encapsulación se produce a todos los niveles de organización: célula, tejido, órgano, aparato, organismo, grupo, población, comunidad, etc.

La expresión diferencial de los genes *Hox* a lo largo del individuo permite compartimentar el cuerpo y distribuir los diferentes objetos (en este caso tejidos y órganos) de forma ordenada a lo largo del animal.

Este concepto tendría que ser tenido en cuenta cuando se elaboran mapas de genomas, sobre todo en los organismos superiores, ya que la información exclusiva de los genes nos da una visión muy parcial de lo que el genoma significa. Aunque ya hay algunos intentos de incluir *enhancers*, *silencers*, promotores, regiones de reconocimiento de factores de transcripción..., que cuando sepamos interpretar correctamente nos darán mucha información (Kolchanov *et al.*, 2000).

Herencia

Es el mecanismo para compartir automáticamente métodos y atributos entre clases y subclases. Esta característica está fuertemente ligada a la reutilización de código. Esto es, el código de cualquiera de las clases existentes puede ser utilizado simplemente con crear una clase derivada de ella. La herencia conduce a una estructura jerárquica de las clases. Esta herencia puede ser múltiple de una clase: pueden derivarse ninguna, una, dos o más clases. Que una clase hija herede las características de una clase madre tampoco tendría mayor importancia, ya que sería una simple copia, si no fuera porque la herencia en programación orientada a objeto permite añadir y redefinir métodos (Macay y Nicolas, 1996). La modificación en un método o dato de la clase madre repercute en la hija si la hija no ha redefinido dicho método o dato. Sin embargo, un cambio en un método o dato de la clase hija no tiene ninguna repercusión en la clase madre[7].

El resultado de este tipo de herencia en los seres vivos se apreciaría claramente en los procesos de diferenciación celular, que durante el desarrollo embrionario observamos. A medida que nos aproximamos a los últimos estadios de diferenciación, las características propias del tejido final se hacen cada vez más patentes, mientras las propias de la célula madre se van desdibujando. La herencia orientada a objeto tiene un significado distinto y más amplio que la herencia en biología Darwinista. Se trata de un concepto más dinámico ya que permite heredar de la clase madre los elementos que se consideran útiles, redefinir elementos de la clase materna e incluso añadir elementos nuevos.

Para apreciar la verdadera versatilidad de este concepto, tal vez sea mejor ilustrarlo con un ejemplo teórico que, tal vez no sea demasiado correcto, pero puede ser aclarador. Supongamos un pez que ha pasado de

ser cartilaginoso a óseo[8]. En este proceso de adaptación a un nuevo tipo de esqueleto hay muchas características que han de cambiar para permitir el crecimiento de las estructuras óseas. Entre ellos conseguir digerir la nueva matriz inorgánica. Desde esta perspectiva, lo primero que hay que hacer es buscar qué tipo celular está afectado y cuál será su «conflicto biológico». En este caso, el tipo celular está afectado será el macrófago y su conflicto «no poder digerir la matriz inorgánica». En este momento, algunos de los macrófagos inmersos en la matriz ósea entrarán en «conflicto activo».

Opciones a explorar, según la intensidad del conflicto

- Modular los niveles de producción de las proteínas en expresión, tanto las propias de la clase macrófago, como las *house-keeping*, mutando promotores y enhancers.
- Hacer réplicas de dichas proteínas, ya sea en microcromosomas o dentro del cromosoma. Esta opción se tomaría si ha habido una resolución parcial del conflicto por parte de la célula al expresar dichas proteínas.
- Introducir mutaciones controladas en las proteínas antes citadas.
- Desdiferenciar la célula hasta el progenitor del macrófago, es decir, monocito.
- Hacer pruebas de expresión y mutación de los genes propios de la clase monocito.
- Generar una nueva clase hija de monocito que en este caso llamaremos clase osteoclasto.
- Probar la incorporación de cualquier otro gen a la definición de clase hija de monocito (osteoclasto).
- Conforme la intensidad del conflicto aumenta, la tasa de mutación que se introduce en los genes también lo hace.
- En este caso, cuando se consigue incorporar una fosfatasa alcalina a la definición de osteoclasto el conflicto se solucionaría y la clase osteoclasto se fijaría.

Otras consideraciones

Si el conflicto sigue sin resolverse, la célula sigue desdiferenciándose tras sucesivos intentos de diferenciación fallidos, en muchos casos acaban con la muerte por apoptosis celular, este mecanismo intenta truncar

la exploración, por parte de la célula, de vías no adaptativas. De forma progresiva, si el conflicto no se resuelve, se evoluciona hacia estadios más primitivos en el desarrollo embrionario. Este proceso de desdiferenciación, muerte y división celular activa es interpretado por los patólogos, en muchos casos, como una tumoración. La frontera entre la célula cancerosas y las células madres es muy delgada (Tannishtha, 2001).

Durante el proceso de intento de adaptación se prueban multitud de proteínas, muchas de las cuales están mutadas, motivo por el cual se pliegan mal. Las chaperonas resultaría entonces tremendamente importantes, no sólo porque ayudaría a plegar las proteínas, sino porque también determinarían qué proteínas se pliegan mal, para así marcarlas por ubiquitinación, señalización que las envía a los lisosomas para ser «recicladas».

Si la célula que entra en «conflicto», pertenece a una clase ya tremendamente diferenciada como las neuronas, las células musculares, los adipocitos, etc., no podrá poner en marcha este mecanismo ya que el alto grado de diferenciación se lo impidiria; tan sólo se producirá una alta tasa de mortalidad celular.

El grado de proliferación celular durante el «conflicto activo» dependería del origen embrionario del tejido. Las células provenientes del endodermo, por ejemplo, presentan una alta tasa de división celular durante el «conflicto activo» dando lugar a tumoraciones.

En los casos en que la tasa de división es muy baja durante el «conflicto activo», lo que se aprecia es la aparición de ulceraciones. Sin embargo, cuando el conflicto se resuelve, si que empezarán a dividirse activamente para expresar los nuevos genes incorporados, si es que los ha habido, y es entonces cuando observamos la aparición de tumores. Si esta célula en «conflicto» se ha desdiferenciado mucho, al expresar sus «adaptaciones» tras la «resolución» producirá tumores en los que se encontrarán tipos celulares diferentes, todos aquellos que se puedan originar a partir del estadio de célula madre en el que se encuentre dicha célula.

Durante el desarrollo embrionario, la diferenciación de las células germinales es anterior a la de endodermo, mesodermo y ectodermo. Esto hace que ante la «resolución» de «conflictos de infertilidad», que afecta a células germinales, tras la desdiferencición de éstas durante el «conflicto activo», se produce un tipo de tumor muy especial: el teratoma. Los teratomas son tumores que presentan las tres láminas embrionarias. Esto es posible debido a que las células germinales desdiferenciadas son pseudozigotos mutados.

Objetos

En programación orientada a objeto, tanto los datos como los métodos se encuentran encapsulados formando objetos. De hecho en programación orientada a objeto tan sólo existen objetos, que actúan a modo de caja negra, y que en realidad las **clases** y **subclases** son definiciones del cómo se ha de crear el objeto.

Por decirlo de alguna manera, la definición de clase sería el código (encapsulado en forma de DNA) presente en el genoma que define la clase monocito. Este código es potencialidad, pero hasta que el DNA no se expresa en forma de proteínas el objeto célula monocítica no existe.

Mensajes

Los mensajes entre objetos son los que originan cambios en el estado del objeto. El objeto receptor de un mensaje debe conocer perfectamente qué ha de hacer tras recibir dicho mensaje, pero cuando emite una respuesta no necesita conocer cómo se desarrolla, simplemente que se está desarrollando.

Una célula pancreática ante el mensaje –concentración elevada de azúcar en sangre– ha de saber producir insulina y liberarla al torrente sanguíneo. Pero le ha de traer sin cuidado si el resto de células saben interpretar que la insulina les da permiso para tomar azúcar del medio. Tan sólo ha de constatar que la concentración de azúcar en sangre baja.

Dicho de otra manera, el «*debugger*» permite relacionar el código fuente del programa (DNA) con el proceso en ejecución (proteína expresada), pudiendo introducir cambios controlados durante la ejecución del programa (la fidelidad de copia de la RNA polimerasa se podría modular). Para ello la tasa de transcripción del DNA ha de ser baja, ya que de otra manera se complicaría mucho la identificación del RNA mensajero que incluye alguna mutación adaptativa.

Ejemplos biológicos de adaptación

Uno de los principios fundamentales de la programación, aunque no aparece en ningún manual, es: **si funciona, no lo toques**. Este consejo, por experiencia os lo puedo asegurar, es tremendamente útil. Resultaría absurdo que la vida no lo hubiera adoptado. A no ser que exista una razón clara para cambiar algo, ya sea por que no funciona o porque lo

hace de forma insuficiente, ninguno de los mecanismos de adaptación facilitada se pondrán en marcha. Por lo tanto, y como premisa para que esta maquinaria actúe, ha de existir un desencadenante al que llamaremos «conflicto biológico». Durante el tiempo que dicho conflicto esté «activo», y dependiendo de la intensidad del conflicto, un tipo u otro de mecanismo adaptativo se pondrán en marcha,.

En biología no existe una solución única para resolver los problemas. Una actividad deficiente de un enzima puede ser resuelta de muchas maneras: duplicando el gen, aumentando la eficiencia de su promotor, aumentando por mutación del gen su actividad enzimática, etc. A continuación se describen algunos ejemplos sobre algunas de las soluciones adaptativas encontradas por ciertos sistemas vivos para solucionar sus «conflictos biológicos».

Generación de soluciones adaptativas in vitro

El ejemplo mejor caracterizado de mecanismo adaptativo con repercusión en el genoma es la resistencia al metotrexato (mtx), uno de los primeros quimioterápicos anticancerígenos administrados por la medicina científica. La célula a la que se administra el metotrexato puede adquirir resistencia mediante mutaciones que cambian la actividad de la enzima dihidrofolato reductasa (DHFR) o amplificando el número de genes estructurales *dhfr*, siendo esta última forma de adaptarse la más frecuente. Fenómenos similares han sido observados en más de 20 genes (Lewin, 1989).

Es decir, el metotrexato sería el desencadenante de un «conflicto biológico activo», ya que al bloquear el metabolismo del ácido fólico el metotrexato impide que la célula se divida por falta de timina para fabricar su DNA. La «resolución» del «conflicto de falta de timina» se logra al conseguir la resistencia tras pasar de 1 copia del gen *dhfr* entre 40 y 400 copias dependiendo de la presión selectiva o sea, la «intensidad del conflicto». De las líneas que adquieren resistencia al metotrexato *mtxr* se pueden distinguir dos clases, las que mantienen la resistencia y forma **estable** y las que son **inestables**.

Las líneas **inestables** presentan cromosomas diminutos dobles (double-minute) en los que se encuentran las copias del gen *dhfr*. Estas líneas, al dividirse en ausencia de metotrexato, suelen ir perdiendo estos microcromosomas.

Las líneas **estables** no pierden su resistencia. En este caso las réplicas del gen *dhfr* se encuentran integrados en el cromosoma junto a la co-

pia original del gen. Pero generalmente esta copia se da sólo en uno de los dos locus[9].

Generalmente la estrategia de generar diminutos dobles es una buena opción para una adaptación temporal. Pero a largo plazo, si se quiere fijar la adaptación, se ha de optar por «encapsular» estos genes en el cromosoma junto a los restantes genes que definen la clase.

Éste es uno de los mecanismos a los que se atribuye la adquisición de resistencia a la quimioterapia de los tumores en pacientes oncológicos. Pero no deja de ser curioso que estos mecanismos aparezcan en células tumorales (Guo, 1999), mientras que no parecen ser comunes en las células consideradas «sanas».

Generación de soluciones adaptativas in vivo *y el papel de ciertos virus que actuarían a modo de «parche del sistema»*

Volviendo a los símiles informáticos. Supongamos por un momento que todos los ordenadores que funcionan bajo windows 98 son como células de un superorganismo, ya que no dejan de ser copias exactas de un mismo programa. El programa ya ha sido distribuido por todo el mundo, pero los usuarios han detectado un problema a la hora de imprimir: cuando dos usuarios intentan imprimir a la vez en una impresora compartida los derechos de los dos usuarios entran en «conflicto» y el ordenador que comparte la impresora se cuelga.

¿Qué estrategia sigue microsoft en estos casos? Evidentemente, contacta con el programador especializado en control de impresión que diseño el módulo de impresión y, tras aplicarle el correctivo oportuno, le hace trabajar a marchas forzadas en una máquina controlada hasta que consigue subsanar el problema. Durante el proceso, probablemente la máquina del programador se colgará en multitud de ocasiones, tendrá que reprogramar parte del sistema y modificar alguna librería. Pero al final del proceso, si todo va bien, conseguirá hacer que todo funcione correctamente. Microsoft ya tiene la solución al problema de impresión, y el programador vuelve a dormir tranquilo. Ahora bien, ¿cómo sabe microsoft qué clientes están teniendo problemas para compartir sus impresoras? La solución más económica es poner en red y a disposición de quien lo quiera un parche para solucionar el problema que tan sólo reinstala la librería modificada. Eso sí, a partir de ahora en las nuevas distribuciones el sistema operativo ya vendrá con la nueva librería.

Traslademos este ejemplo al ámbito de la biología y veamos tres ejemplos, uno en humanos, otro en aves de corral y un tercero en insec-

tos. Cada uno de estos ejemplos representa un problema adaptativo concreto.

1. Supongamos que tenemos un individuo que está bajo un «conflicto biológico» similar al que analizamos en los cultivos celulares. Por ejemplo, un enfermo al que se le ha diagnosticado un SIDA, que sigue su terapia rutinaria con Dacortin, Azt, Aciclovir, Septrim o Bactrim, y demás sustancias. De todos estos preparados, para este ejemplo nos centraremos en el Septrim (Cotrimoxazol). Los dos componentes principales de este preparado son el trimetoprim y el sulfametoxazol (TMP/SMX). Ambos bloquean de dos formas distintas la síntesis de timina, impidiendo la división celular por imposibilidad de sintetizar DNA. Si realizamos los paralelismos oportunos con el símil informático, el sistema biológico encargaría la tarea de encontrar una solución no a todas y cada una de las células, sino a unas células especializadas. Ya hemos visto que las células cancerosas son especialmente adaptables. A pesar de lo que pudiera derivarse de una visión Darwinista, los tumores en los pacientes diagnosticados de SIDA no son mucho más comunes que en la población *normal*. Esto debería chocarnos, ya que se trata de paciente con las *defensas* bajas. Hay alguna excepción, ciertos tipos de cáncer son muy comunes en enfermos a los que se les ha diagnosticado un SIDA, como son los linfomas, los sarcomas y, en mujeres, el cáncer de cuello de útero. Como ya se vio en el ejemplo del metotrexato, la célula tumoral, a modo de programador experto, es perfectamente capaz de adaptarse y resolver el problema de la deficiencia de timina. Pero, ¿es capaz de exportar esta capacidad al resto de las células con el mismo problema a modo de «parche del sistema»? Los sarcomas de kaposi han sido los más estudiados y algunos de los virus que estos producen, como el KSHV (Kaposi sarcoma-associated herpesvirus), ha sido secuenciado (Russo *et al*, 1996). ¿Qué genes debería contener este virus, para que realmente se comportase como un «parche del sistema»? Por un lado, los genes propios del virus que actuarían a modo de soporte informático, pero además deberían incluir toda la batería de genes adaptativos que la célula tumoral se ha encargado de seleccionar y procesar para solucionar el «conflicto biológico» en curso. ¿Qué genes no propiamente virales se deberían encontrar? Deberían estar aquellos que codifiquen para alguna vía de la síntesis de timina o directamente para la incorporación de timidina. A continuación se detallan algunos de los genes presentes en el genoma del virus KSHV (Russo *et al*, 1996): Timidilato sintasa (metila a partir metilentetrahidrofolato, el dUMP a dTMP liberando dihidrofolato), dihidrofolato reductasa (regenera el dihidrofolato a tetrahidrofolato), dUTPasa (defosforila del dUTP a dUMP), Timidilato kinasa (le permite incorporar timidina), ribonucleótido reductasas, etc.

Al menos en apariencia, este virus ha recopilado una batería de genes que pueden ayudar a superar una deficiencia severa de timina. Este virus podría ser de reciente creación, desarrollándose paralelamente a la administración de este tipo de medicación, pero también es posible que evolucionase al cabo de los años tras múltiples conflictos de deficiencia de ácido fólico. Sea como fuere, este virus se asemeja mucho a lo que se esperaría que fuese un parche biológico del sistema para subsanar una deficiencia de timina.

2. En el caso de un ave de corral, su «conflicto biológico» derivaría de las condiciones en las que viven debido a los criterios de productividad y de reducción de costes de producción que se aplican actualmente a las granjas de cría de pollos. En el genoma de uno de los virus que infecta a estos animales (Afonso, 2000) encontramos genes que podrían ser resultado de una respuesta adaptativa a este tipo de problema. Encontramos genes de detoxificación (Glutatión peroxidasa) que resultarían muy útiles para animales que consumen piensos de baja calidad y están sometidos fuertes condiciones de estrés. También encontramos resistencias a antibióticos como la rafampicina, que por ser un antibiótico barato no sería de extrañar que se suministrase a los pollos.

3. El tercer ejemplo que se muestra es el de un virus de saltamontes (Afonso, 1999), los cuales podrían estar enfrentándose a nuevos «conflictos biológicos» derivados del uso generalizado de los pesticidas. Los pesticidas presentan varios frentes de acción, entre ellos provocar lesiones en el DNA. La presencia en el virus de varios genes de reparación del daño en el DNA pueden ser útiles para el hospedador del virus (uracil DNA glicosilasa, AP endonucleassa, DNA polymerasa ß, topoisomerasa I y NAD+ dependiente DNA ligasa). Además aparecen en el virus inhibidores de la apoptosis celular, muy útiles para la supervivencia de las células del hospedador del virus, ya que el daño del DNA puede desencadenar por si solo la apoptosis celular.

Distribución de los «parches del sistema»

Virus como poliovirus, coxsackievirus, echovirus y enterovirus en general también podrían ser unos buenos candidatos para funcionar a modo de «parche del sistema». Además estos virus presentan una altísima tasa de recombinación en los genes que no codifican para su propia cápside (Santti, 1999). Esta característica podría contribuir a crear un conjunto de genes de procedencia diversa esperando que alguien los adopte para su acerbo genético.

La transmisión horizontal de genes sería un mecanismo mucho más generalizado de lo que se consideraba hasta ahora. Así parecen confirmarlo algunos modelos matemáticos (Qian, 2001). Qian y sus colaboradores, partiendo de los genomas completos secuenciados hasta el momento, calcularon la distribución de familias, superfamilias y plegamientos (*folds*) en los diferentes genomas secuenciados. Siguiendo el modelo por ellos expuesto, la distribución de dichas frecuencias se comportaría como una función exponencial si el componente principal de la variabilidad genética viniese de la duplicación interna de genes. Sin embargo, la adopción de proteínas externas con otros plegamientos, durante un largo periodo de evolución, daría lugar a una distribución potencial. La conclusión final a la que se llega en el estudio es que la distribución tanto de las familias como las superfamilias y los plegamientos de proteínas siguen una distribución potencial. En otras palabras, que tanto la duplicación como la transmisión horizontal de genes juegan un papel crucial en los mecanismos de evolución celular.

Como ya se ha adelantado al explicar el «*debugger* celular», la activación del mismo facilitaría a adopción de genes externos, como serían los que arrastran los virus mencionados.

Los genes exportados por estos virus pueden fijarse en las células somáticas propias, en las células de otros individuos de la especie, o incluso en las células de los individuos de otras especies (Staeheli, 2000). Incluso, como demuestra la enorme cantidad de secuencias de origen vírico encontradas durante la secuenciación del genoma humano, acabar fijándose en las células germinales, tal vez por la acción de los transposones.

No todos los virus son «parches del sistema»

La mayoría de los virus no presentan genes ajenos en su secuencia, ya que las características de su cápside no permiten empaquetar más genes que esos. Estos virus no podrían funcionar a modo de «parche del sistema».

Incluso existen virus que se comportan como auténticos depredadores. Por ejemplo, el virus de la rabia, al atacar el sistema nervioso central de su víctima, produce lesiones que estimulan regiones del cerebro que exacerban la conducta fóbica y agresiva del individuo infectado. En estas circunstancias, el individuo infectado interpreta cualquier situación como una agresión y tiende a morder cualquier otro animal que se le aproxime. Dado que una pequeña parte de los virus de la rabia se re-

producen en las glándulas salivares, esto propicia el inicio del siguiente ciclo de infección.

Recapitulación

La naturaleza de los seres vivos es tremendamente compleja y está llena de matices, por este motivo es tan difícil extraer leyes generales que expliquen su funcionamiento. En el texto se ha intentado esbozar parte de un mecanismo de «adaptación facilitada» que pretendidamente y de forma automática actúa como director de la adaptación celular, promoviendo modificaciones «adaptativas» del genoma de la célula. Las modificaciones inducidas sobre el genoma se producirían por un mecanismo celular bautizado como «*debugger* celular» que, respetando la arquitectura orientada a objeto atribuida a la organización de los seres vivos, exploraría las posibilidades adaptativas de la célula. Las modificaciones más comunes del genoma serían: mutaciones puntuales de genes y/o promotores, duplicación de genes, inserciones de genes ajenos y recombianción con genes propios o ajenos, *trans-splicing*, *splicing* alternativo....

La propagación horizontal de estos genes «adaptativos» se realizaría mediante virus.

Agradecimientos

«A ti, por tomarte la molestia de leer este texto. A toda esa maravillosa gente que me rodea, ha sabido escucharme y tanto me ha enseñado, incluso cuando, en algunos casos, mis ideas difirieran diametralmente de las suyas».

Notas

[1] Mutación que a pesar de dar lugar a un cambio en la secuencia del triplete de DNA, debido al uso de codón propio de dicho organismo, no se traducirá en un cambio de aminoácido en la proteína codificada.

[2] Traducido a términos biológicos, si una célula contase con una utilidad tipo «*debugger*», le permitiría probar diferentes fenotipos sin necesidad de cambiar su genotipo.

[3] Las cepas trp- crecen sobre la placa hasta agotar el triptófano del medio. Hasta ese momento, aparecen como una colonia única en fase estacionaria. Si se produce una reversión en el conjunto de las células que forman la colonia, éstas crecerán sobre la colonia, apareciendo una papila. Contar las papilas permite establecer la tasa de reversión de la mutación trp– que se está estudiando.

[4] Dentro del modelo de «adaptación facilitada» la presencia de la proteína p53 estaría indicando que la célula está en «conflicto activo» y que el mecanismo de «*debugger* celular» se ha disparado.

[5] Estos conceptos se detallan más adelante en el texto.

[6] Desde el punto de vista biológico, esta característica de la programación orientada a objeto, que permite atacar el diseño de una aplicación o una función con un planteamiento tan difuso como ¿de qué trata el programa?, harían de este tipo de programación un sistema ideal como propuesta, para sistemas vivos adaptativos.

[7] Esta estructura jerárquica y la encapsulación de las clases es especialmente interesante para modelos de evolución adaptativa en seres vivos, ya que protegerían las células madres de los daños derivados de los intentos fallidos de adaptación de sus células hijas.

[8] Algunos cambios aunque son respuestas adaptativas a un «conflicto biológico» concreto, generan una cascada de nuevos «conflictos» que al ser resueltos en cadena, dan lugar a los saltos evolutivos que apreciamos en el registro fósil.

[9] La estrategia de introducir los cambios en sólo uno de los locus dejando el otro intacto, es un excelente mecanismo de seguridad que garantiza que si el cambio, pretendidamente adaptativo, no funciona o genera problemas, al menos se contara con un original.

Bibliografía

1. AFONSO, C. L., TULMAN, E. R., LU, Z., ZSAK, L., KUTISH, G. F. y ROCK, D. L., «The Genome of *Melanoplus sanguinipes* Entomopoxvirus», **73**, 533-552, 1999.
2. AFONSO, C. L., TULMAN, E. R., LU, Z., ZSAK, L., KUTISH, G. F. y ROCK, D. L., «The Genome of Fowlpox Virus», **74**, 3815-3831, 2000.
3. ALBOR, A., KAKU, S., y KULESZ-MARTIN, M., «Wild-type and mutant forms of p53 activate human topoisomerase I: a possible mecanism for gain of function in mutants», Cancer Res., **58**, 2091-2094, 1998.
4. ALEXANDER, L., DENEKAMP, L., KNAPP, A., AUERBACH, M. R, DAMANIA, B. y DESROSIERS, R. C., «The primary Squence of Rhesus Monkey Rhadinovirus Isolate 26-95: Sequence Similarities to Kaposi´s Sarcoma-Associated Herpesvirus and Rhesus Monkey Rhadinovirus Isolate 17577» J. Virol. 3388-3398, Apr. 2000.
5. BEETHAM, P. R., KIPP, P. B., SAWYCKY, X. L., ARNTZEN, C. J. y MAY, G. D., «A tool for functional plant genomincs: Chimeric RNA/DNA oligonucleotides cause in vivo gene-specific mutation», Proc. Natl. Acad. Sci. USA, **96**, 8774-8778, 1999.
6. CEBALLOS, F. J., «Programación orientada a objetos con C++» Ed. Rama, 1997.
7. ELLIS, H. M., YU, D., DITIZIO, T. y COURT, D. L., «High efficiency mutagenesis, repair, and engineering of chromosomal DNA using single-stranded oligonucleotides», Proc. Natl. Acad. Sci. USA, **98**, 6742-6746, 2001.
8. GUO, W., HEALEY, J. H., MEYERS, P. A., LANDANYI, M., HUVOS, A. G., BERTINO, J. R. y GORLICK, R., «Mechanims of Methotrexate Resistence in Osteosarcoma» Clin. Cancer Res., **5**, 621-627,1999.
9. HALL, B. H., «Adaptative evolution that requires multiple spontaneous mutations: Mutations involving base substitutions». Proc. Natl. Acad. Sci. USA, **88**, 5882-5886, 1991.
10. HERRERAa, V., «De la Enfermedad a la Vida», Ed. Club de Autores, 1999.

11. HERRERAb, V., «Entender la Vida Comprender la Enfermedad», Ed. Club de Autores, 1999.
12. KOLCHANOV, N. A. *et al.* «Transcription Regulatory Regions Database (TRRD) its status in 2000» , Nucleic Acids Res., **28**, 298-301.
13. LANE, D., «How Cells Choose to Die», Nature, **414**, 25-26.
14. LEWIN, B., «Genes». 1989, Ed. Reverté.
15. LJUNGMAN, M. y PAULSEN, M. T., «The Cyclin-Dependent Kinase Inhibitor Roscovitine Inhibits RNA synthesis and Triggers Nuclear Accumulation of p53 That Is Unmadified at Ser15 and Lys382», Mol Pharmacol, **60**,785-789,2001.
16. LIU, L., MICHAEL, C., KMIEC, R., y KMIEC, E. B., «In vivo gene repair of point and frameshitf mutations directed by chimeric RNA/DNA oligonucleotides and modified single-stranded oligonucleotides». Nucleic Acid Res., **29**, 4238-4250, 2001.
17. MACAY, J. F. y NICOLAS, C, «Programación Java» Ed. Eyrolles, 1996.
18. VAN OIJEN, M. G. y SLOOTWEG, P. J., «Gain-of-Funtion Mutations in the Tumor Supressor Gene p53» Clin. Cancer Res. **6**, 2138-2145, 2000.
19. QIAN, J., LUSCOMBE, N. M. y GERSTEIN, M., «Protein Family and Fold Occurence in Genome: Power-Law Behavior and Evolutionary Model», J. Mol. Biol., **313**, 673-681.
20. ROJO, F., «Repression of Transcription Initiation in Bacteria». J. Bacteriol, 2987-2991, 1999.
21. RUSSO, J. J., BOHENZKY, R. A., CHEIN, M.C., CHEN, J., YAN, M., MADDALENA, D., PARRY, J. P., PERUZZI, D., EDELMAN, I. S., CHANG, Y. y MOORE, P. S. «Nucleotide sequence of the Kaposi sarcoma-associated herpesvirus (HHV8)». Proc. Natl. Acad. Sci. USA **93**, 14862-14867, 1991.
22. SANTTI, E., HYYPIÄ, T., KINNUNEN, L. y SALMINEN, M., «Evidence of Recombination among Enteroviruses», J. Virol., **73**, 8741-8749, 1999.
23. STAEHELI, P., SAUDER, C., HAUSMANN, J., EHRENSPERGER, F. y SCHWEMMLE, M., «Epidemiology of *Borna* disease virus», Journal of General Virology, **81**, 2123-2135, 2000.
24. VISPÉ, S., YUNG, T. M. C., RITCHOT, J., SERIZAWA, H. y SATOH, M. S., «A cellular defense pathway regulating transcription through poly(ADP-ribosil)ation in response to DNA damage. Proc. Natl. Acad. Sci. USA **97**, 9886-9891, 2000.
25. WILLIAMSON, P. G., «Speciation in mollucs from Turkana Basin». Nature, 659-663, 1983.
26. TANNISHTHA, R., MORRISON, S. J., CLARKE, M. F. y WEISSMAN, I. L. «Stem cells, cancer, and cancer stem cells», Nature, 414, 105-111, 2001.
27. TOPALIAN, S. L., KANEKO, S., GONZALES, M. I., BOND, G. L., WARD, Y. y MANLEY, J. L. «Identification and functional characterization of neo-poly(a) polymerase, an RNA processing enzyme overexpressed in human tumors», Mol Cell Biol, **21**, 5614-5623, 2001.