Sistemas Informáticos
Curso 2005-2006

Desarrollo de una plataforma de análisis de datos en bioinformática basada en servicios web

Gerardo Hernández Campos
Manuel Domingo Mora Martínez
Jesús Patiño Gómez

Dirigido por:
Prof. Alberto Pascual Montano
Dpto. Arquitectura de Computadores y Automática

Facultad de Informática
Universidad Complutense de Madrid

Prefacio

Este proyecto surge ante la necesidad de una plataforma no dependiente del modelo tradicional del uso de la web. Las aplicaciones tradicionales de bioinformática presentan esta dependencia, haciendo necesaria la intervención humana cuando se quiere hacer uso de las mismas.

Es por ello que se cree necesario el desarrollo de una aplicación que haga un uso más eficiente de los recursos proporcionados por la web, y surge la idea de la creación del programa desarrollado basado en servicios web.

El servicio web ha sido desarrollado en C++, haciendo uso del toolkit de generación de código gSOAP (ver sección 5).

La plataforma elegida fue Eng*e*ne [10] (ver sección 2), por tratarse de una herramienta disponible de forma gratuita y la cuál no tiene ningún ánimo de lucro. Se compone de herramientas para visualización y procesamiento de grandes conjuntos de datos genéticos, entre otros.

This project arises from the necessity of a independent platform from the traditional model from the use from the Web. The traditional applications of bioinformatic present this dependency, doing necessary the human intervention when it is wanted to make use of them.

For this reason it is believed to be necessary the development of an application that makes a more efficient use of the resources provided by the Web, and the idea of the developed program based on web service technology arises

The web service has been developed in C++, doing use of toolkit of code generation gSOAP (to see section 5).

The chosen platform was Engene [10] (to see section 2), for being a tool available free and which does not have any spirit of profit. It is composed of tools for visualization and processing of great genetic data sets, among others.

Palabras clave

Bioinformática, servicios web, Engene, gSOAP, WSDL, XML, DIME, GTK, GLADE.

Autorización

Los alumnos Gerardo Hernández Campos, Jesús Patiño Gómez y Manuel Domingo Mora Martínez de la Facultad de Informática de la Universidad Complutense de Madrid, autorizan a la Universidad Complutense a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la propia memoria, como el código, la documentación y/o el prototipo desarrollado.

Fdo. Gerardo Hernández Campos

Fdo. Jesús Patiño Gómez

Fdo. Manuel Domingo Mora Martínez

Índice de contenido

1. Introducción a la Bioinformática

1.1. ¿Qué es la bioinformática?

La secuenciación de proteínas tuvo unos lentos orígenes, comenzando en la década de los '50 con la elucidación de la primera secuencia completa, la de la hormona peptídico insulina [5].

Al inicio de la década de los '60 se obtuvo la secuencia para la primera enzima, la de la ribonucleasa. También en esta década Margaret Dayhoff incluyó todas las secuencias de proteínas conocidas hasta la fecha en *Atlas of Protein Sequence and Structure.* Podría decirse que fue la primera base de datos biológicos. El cual evolucionó después en el PIR.

En la década de los '80 el número estimado de las secuencias conocidas era de 1500. En 1982 el laboratorio europeo de biología molecular (EMBL) desarrolla una base de datos de secuencias de ADN, que junto con GenBank, facilitan una nueva fase en la búsqueda de información de secuencias. El nacimiento de EMBL, que fue la primera base de datos que se creo sobre secuencias de ADN y ARN, y de NCBI, perteneciente a la Library National of Medicine perteneciente a su vez al National Institutes of Health, contribuye a escribir en un formato más adecuado para su uso en los ordenadores los datos aparecidos en las diferentes publicaciones sobre secuenciación de ADN. A partir de esta década comenzó a existir una colaboración entre las tres bases de datos más importantes, EMBL, NCBI (a través de GenBank) y Dna Databank of Japan (DDBJ), las cuales son actualizadas de manera simultanea al introducir nueva información en cualquiera de ellas [1].

Actualmente el crecimiento de los datos almacenados en estas bases de datos presenta un crecimiento exponencial, los cuales duplican su tamaño cada 15 meses. Para el tratamiento de esta ingente información, siendo millones de secuencias de bases, son necesarias herramientas informáticas para su tratamiento de manera eficiente. Aquí es donde juega un papel importante la bioinformática con su capacidad de comparar y relacionar esta información proveniente de fuentes biológicas, para alcanzar respuestas que no podrían obtenerse de manera directa dada la complejidad de su tratamiento.

Inicialmente la obtención de las secuencias se hacía de manera manual, mediante un proceso llamado degradación-dansalición de Edman [5] , pero con la introducción de los secuenciadores automáticos se consiguió un gran aumento en el número de proteínas secuenciadas.

Los grandes volúmenes de información generados a partir de la utilización de las técnicas rápidas de secuenciación de ADN requieren el uso de ordenadores para su tratamiento, De esta forma surgió la bioinformática como una forma de unir todas las aplicaciones informáticas necesarias para las ciencias biológicas [5]. Por tanto la definición de bioinformática es muy amplia, en el sentido de que engloba tantas áreas distintas que se hace difícil dar una definición precisa. Podríamos definirla como una disciplina en la cual intervienen múltiples disciplinas tales como la biología, la informática y los medios de comunicación facilitados por las tecnologías de la información para tratar y analizar datos de origen biológico.

1.2. Aplicaciones bioinformáticas

La mayor parte de las aplicaciones existentes hacen uso de las bases de datos biológicos disponibles. Estos datos son usados de manera directa, para de esta forma obtener similitud entre secuencias o de manera indirecta para de este modo obtener reglas que permitan predecir propiedades en secuencias no conocidas.

Una de las aplicaciones más utilizadas en bioinformática son las relacionadas precisamente con el procesamiento de secuencias de ADN y proteínas. A modo de ilustración incluiremos en esta memoria la descripción de alguna de ellas:

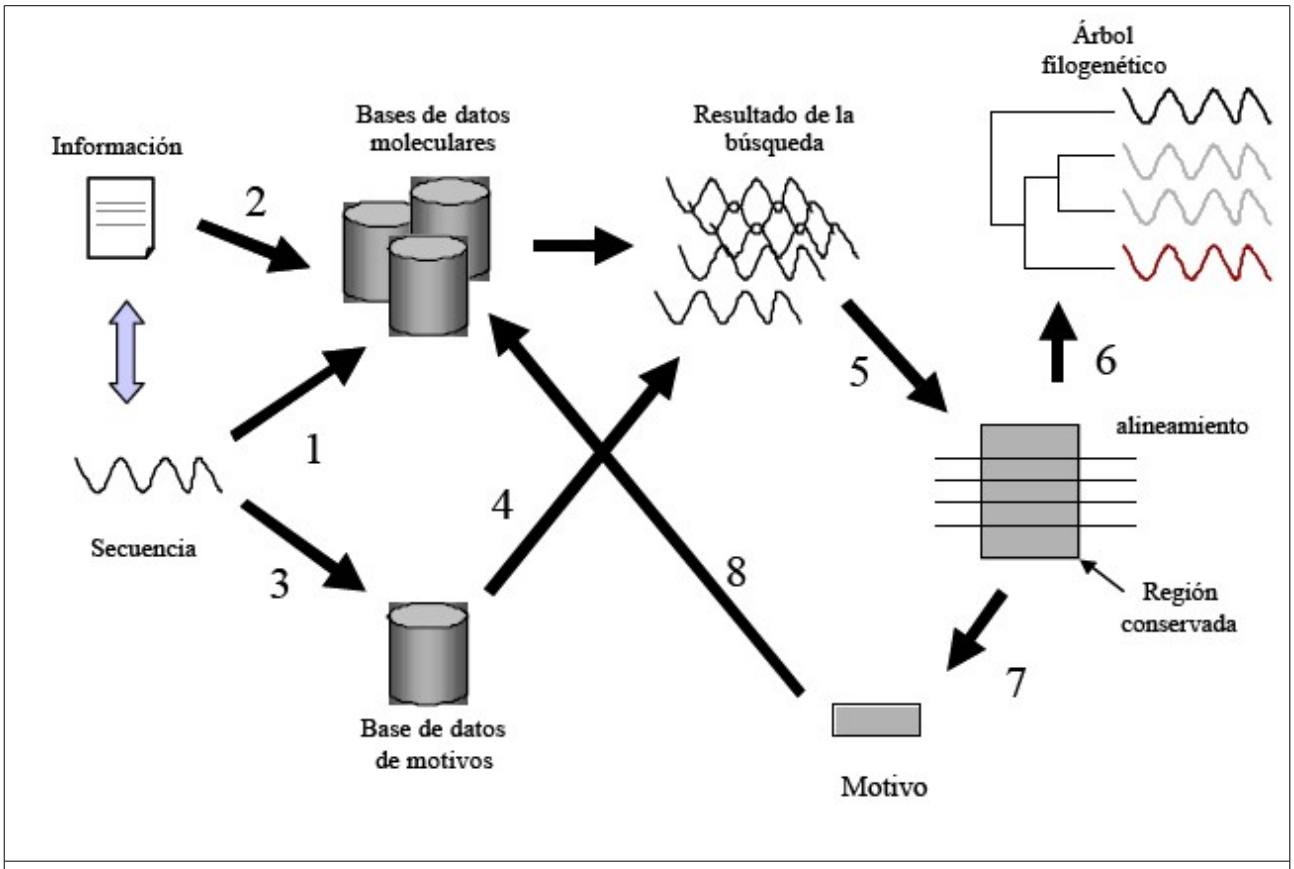

Fig. 1.1. Aplicaciones bioinformáticas
1._ Se obtienen secuencias similares a la secuencia inicial. Ejemplo: FASTA o BLAST.
2._ Se obtienen propiedades de similitud de la secuencia inicial. Ejemplo: SRS.
3._ Se buscan motivos funcionales o estructurales en la secuencia inicial
4._ Se obtienen secuencias similares a la inicial
5._ Se alinean las secuencias. Ejemplo: CLUSTAL.
6_ Se obtiene el árbol filogenético Ejemplo: PHYLIP.
7._ Se obtiene un motivo característico al hacer el alineamiento.
8._ Se usa el motivo para buscar nuevas secuencias. Ejemplo: HMMER.

En la figura 1.1. podemos ver las etapas más comunes en las aplicaciones bioinformáticas relacionadas con el análisis de secuencias. Se parte de una secuencia inicial de la cual se quiere conocer secuencias similares (etapa 1), o que compartan alguna característica con ella (etapa 2). También a partir de la secuencia inicial se pueden obtener motivos estructurales (etapa 3). Esto se hace para poder realizar búsquedas de nuevas secuencias a partir de estos motivos que previamente no se conocían. Tanto a partir de las bases de datos moleculares como a partir de las bases de datos de motivos, se obtienen secuencias similares a la inicial (etapa 4). Una vez que se dispone de las secuencias similares se puede proceder a realizar un alineamiento, sencillo o múltiple (etapa 5), para de esta forma conseguir árboles filogenéticos (etapa 6) o la obtención de motivos del alineamiento (etapa 7). Estos motivos pueden ser utilizados para la búsqueda de nuevas secuencias similares (etapa 8).

1.2.1. Búsqueda de secuencias similares por alineamiento (Blast y Fasta)

Blast y Fasta realizan búsqueda de secuencias similares. Ésta se realiza alienando la secuencia inicial con las contenidas en las bases de datos, para de este modo obtener las más similares. Se hace uso de métodos de programación dinámica para obtener el alineamiento óptimo. Éste se obtiene haciendo uso de una matriz donde se dota de distinta puntuación a los pares de aminoácidos (o nucleótidos) presentes en la secuencia y que quedan enfrentados en el alineamiento. También se hacen unos de penalizaciones cuando se encuentran deleciones donde no hay alineamiento.

Fasta busca k-tuplas iguales, y de esta forma obtiene fragmentos de longitud k, pero pierde información de fragmentos más pequeños. Sin embargo Blast busca fragmentos que puedan producir un emparejamiento positivo, descartando aquellos que produzcan un emparejamiento de menor probabilidad.

Blast puede ser usado desde la dirección http://www.ncbi.nlm.nih.gov/blast/ (figura 1.2.)

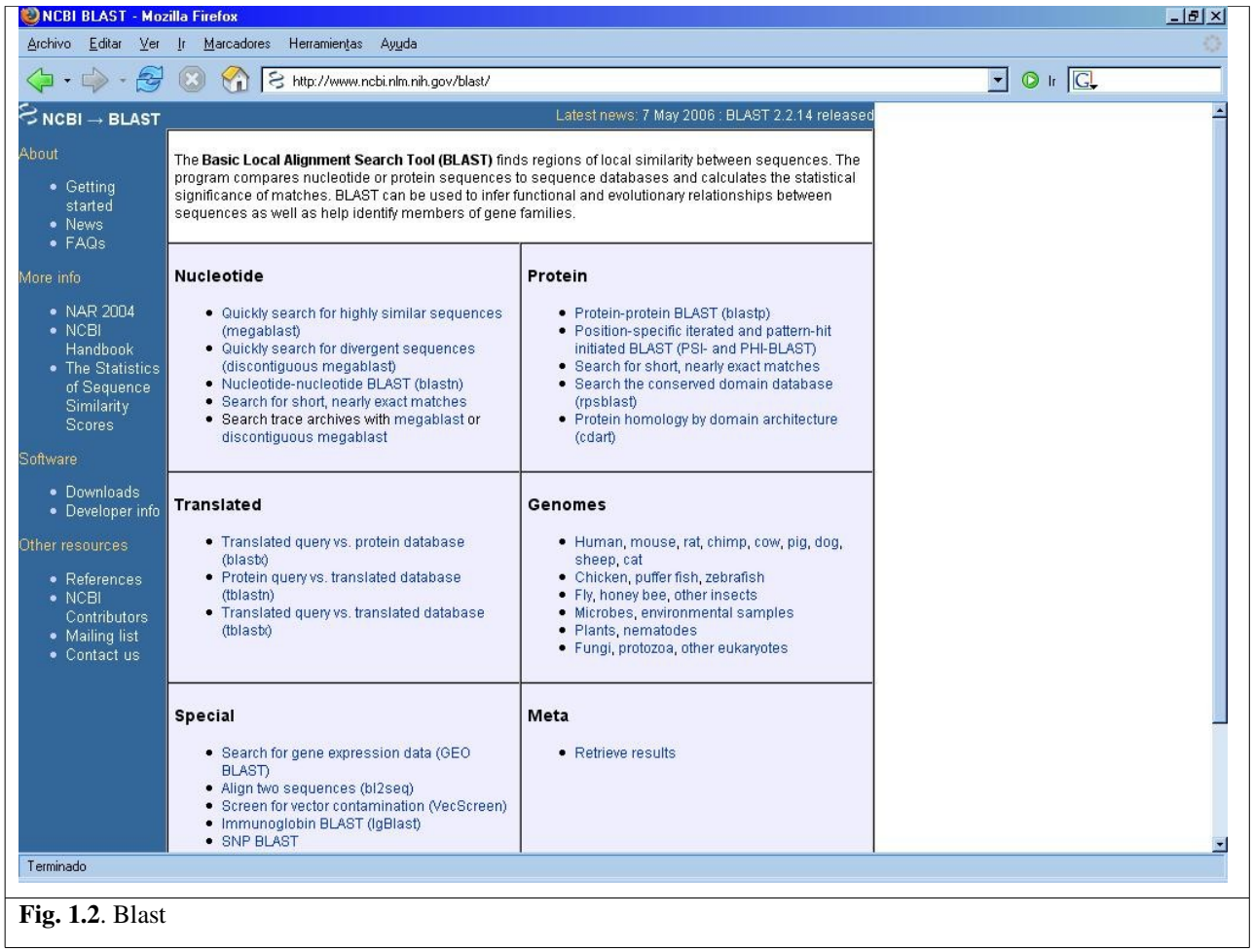

Fig. 1.2. Blast

Una vez introducida la secuencia, Blast devuelve una lista de secuencias ordenadas según el grado de similitud con la inicial.

1.2.2. SRS

SRS (*Sequence Retrieval System*) se utiliza para realizar búsquedas en bases de datos. Su servidor principal se encuentra ubicado en EBI.

En estas bases de datos se pueden realizar búsquedas tanto de secuencias como motivos, códigos genéticos, estructuras de proteínas (en 2D o 3D), mapas genéticos, etc. También contienen resultados de búsquedas de Blast, Fasta, alineamiento, etc.

1.2.3. Alineamiento múltiple: CLUSTAL

CLUSTAL realiza alineamiento múltiple de secuencias homólogas a la secuencia inicial.

Para utilizar CLUSTALW, debemos introducir la dirección http://www.ebi.ac.uk/clustalw/ (fig 1.3.)

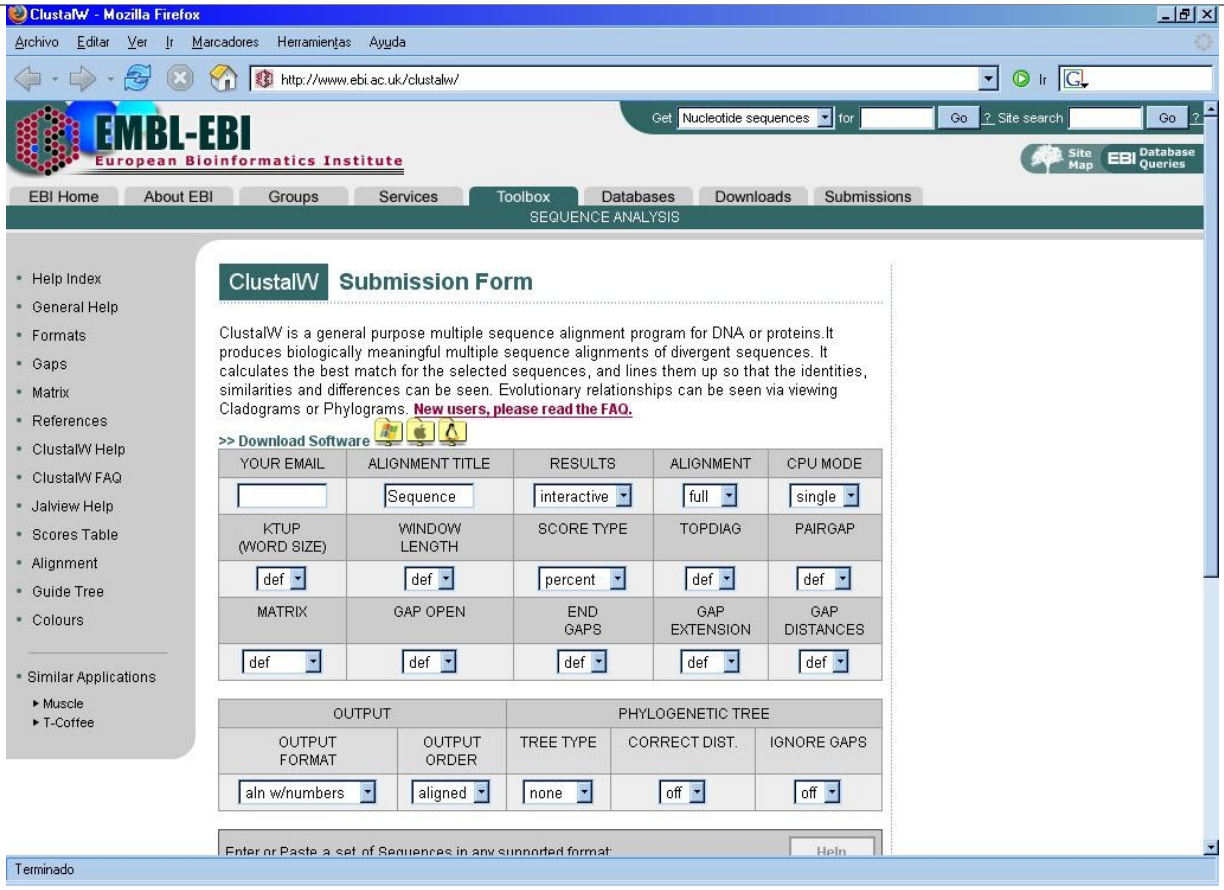

Fig. 1.3. CLUSTALW

La cual una vez introducidas las secuencias y definidos los parámetros de entrada, nos devuelve el árbol filogenético y el alineamiento múltiple de las secuencias.

1.2.4. National Center for Biotechnology Information (NCBI)

En 1988 se creo NCBI, el cual forma parte de NIH (Library National of Medicine of Health). Podemos acceder a sus recursos a través de la siguiente dirección: http://www.ncbi.nlm.nih.gov (fig. 1.4.).
NCBI provee diferentes bases de datos de acceso público. Estas bases de datos son de dos tipos:

- Bibliográficas como por ejemplo PubMed, accesible desde http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?DB=pubmed , que proporciona información biomédica.

- Moleculares como por ejemplo GenBank, accesible desde http://www.ncbi.nlm.nih.gov/Genbank/ y permite la consulta a bases de datos de secuencias de ADN.

Por último se puede mencionar también el desarrollo de software para el análisis de genoma.

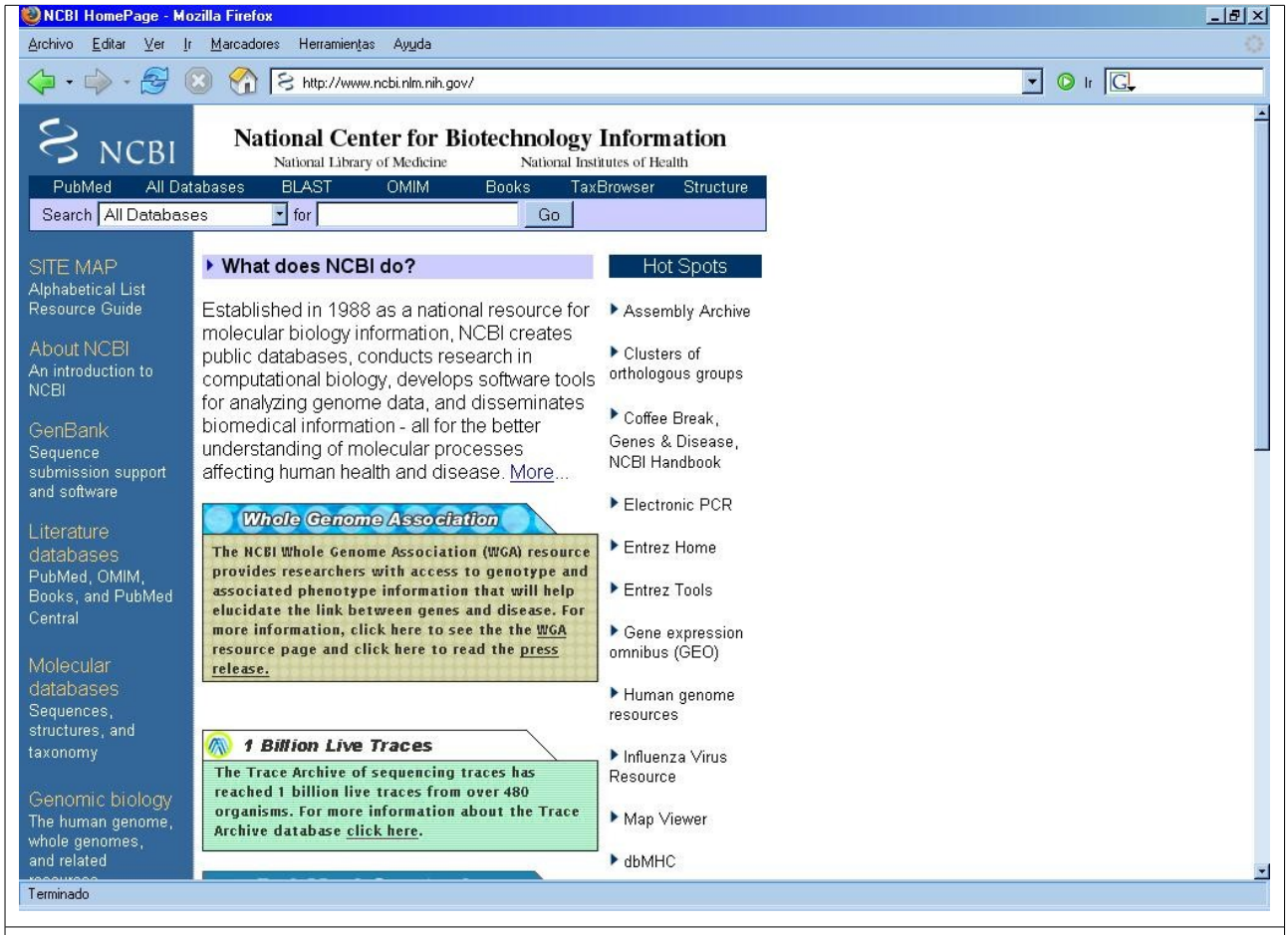

Fig. 1.4. NCBI

1.2.5. Expasy

El propósito de Expasy es de facilitar la búsqueda de secuencias de proteínas, de nuevo a través de Internet. Para ello hace uso de la base de datos SWISS-PROT.

Para poder realizar una búsqueda debemos ir a la dirección http://www.expasy.org/sprot/ , la cual nos muestra la página de la figura 1.3.

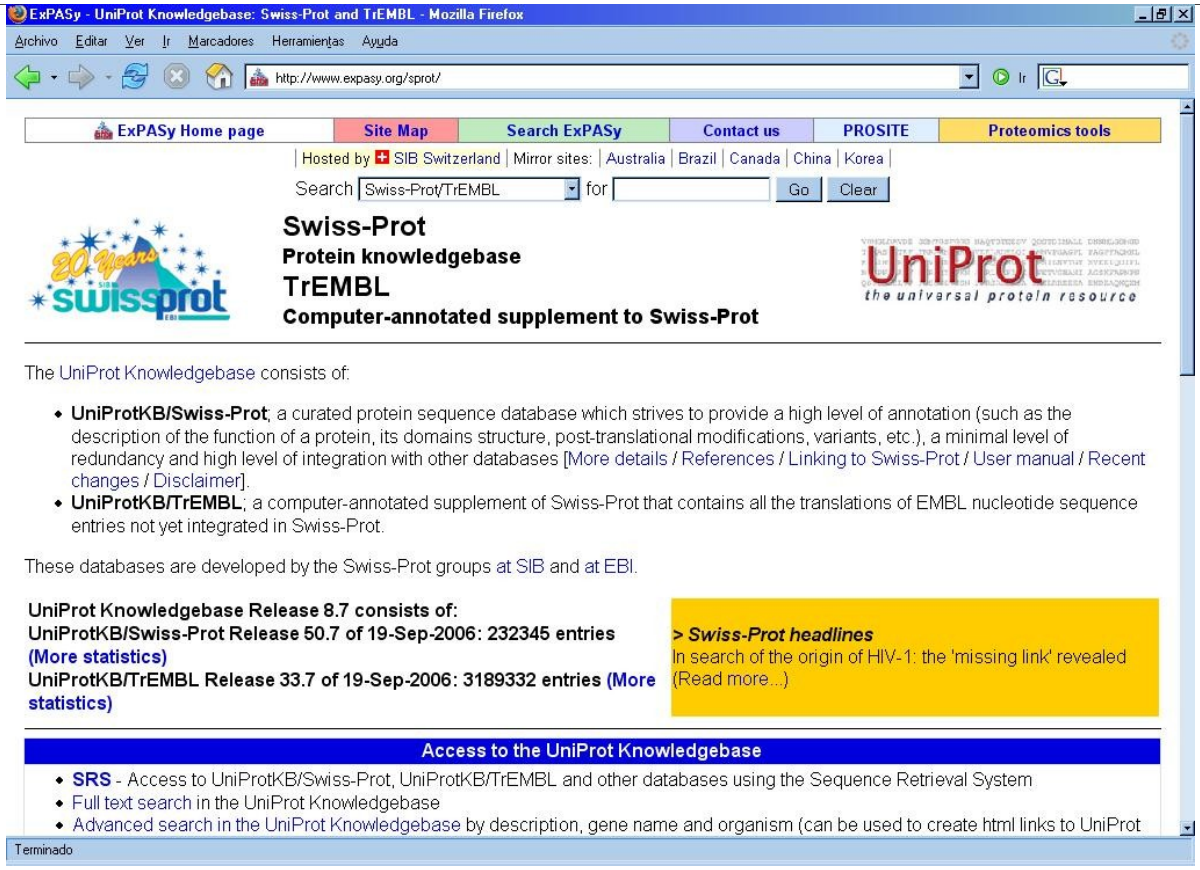

Fig. 1.5. SWISS-PROT

Al introducir el nombre de la proteína nos devuelve una página con las características que se han introducido de dicha proteína en la base de datos. Estas características son tales como el nombre SWISSPROT, el número de acceso, sinónimos, bibliografía y la secuencia de la proteína.

Una de las características de SWISSPROT es la de obtener la secuencia de la proteína en formato FASTA, para de esta forma poder trabajar con ella en los programas de análisis de secuencias, como BLAST o CLUSTALW.

1.2.6 EBI

El Instituto Europeo de Bioinformática (EBI) ofrece bases de datos de ácidos nucleicos, secuencias de proteínas y de estructuras moleculares. Puede ser consultado en la siguiente dirección: http://www.ebi.ac.uk/ (fig 1.6)

Fig 1.6. EBI

Uno de los opciones ofrecidos por EBI es la utilización de web services (fig. 1.7.)

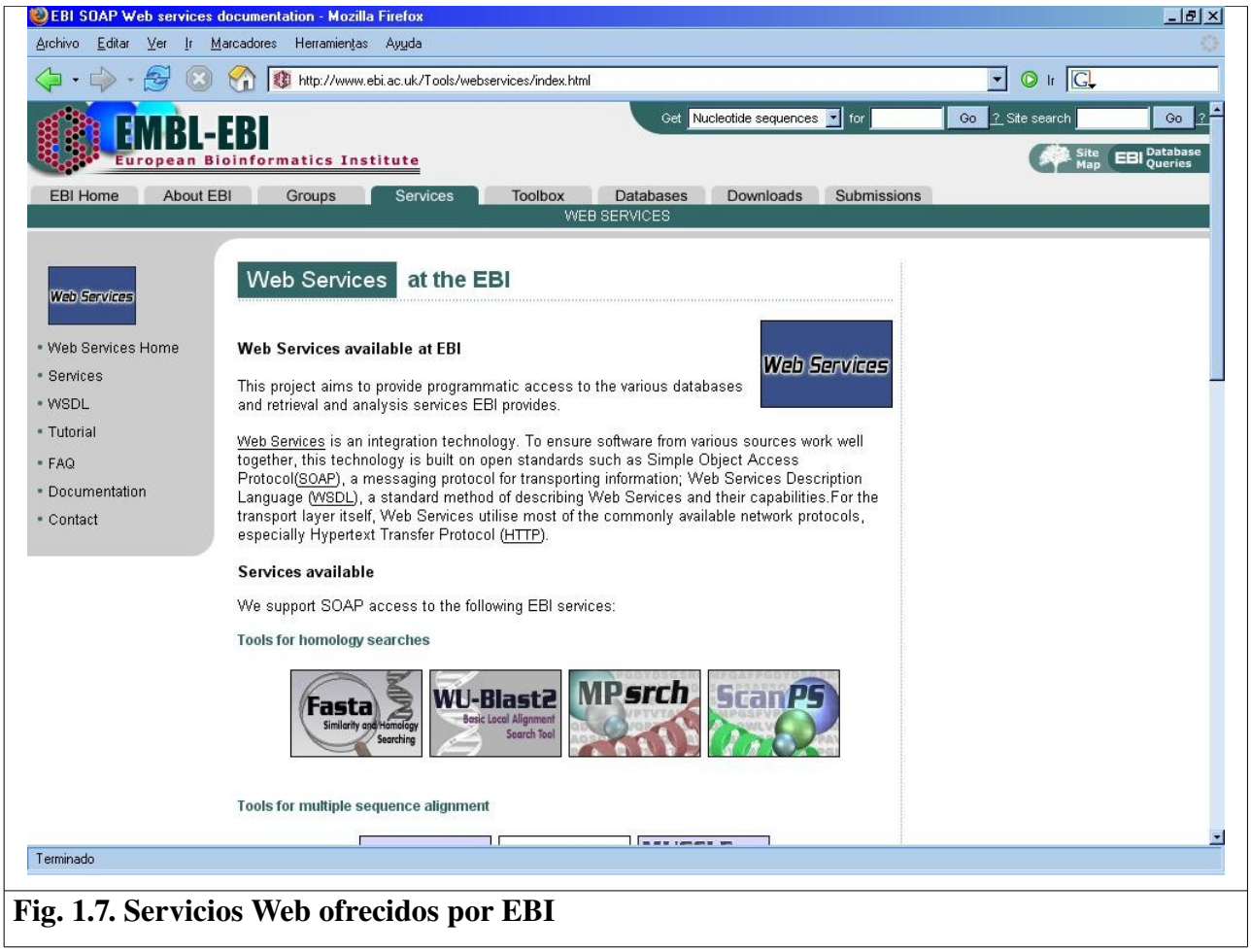

Fig. 1.7. Servicios Web ofrecidos por EBI

Entro los servicios web ofrecidos por EBI, se pueden destacar los siguientes:

- Búsqueda por homología:
    1. WSBlast, realizan una comparación de la secuencia introducida con las existentes en sus bases de datos y devuelve las secuencias con mayor similitud.
    2. WSFasta, realizan una comparación de la secuencia introducida con las existentes en sus bases de datos y devuelve las secuencias con mayor similitud. La diferencia con WSBlast es que WSFasta no permite ambigüedad en las comparaciones y si permite huecos en las regiones similares.
- Alineamiento múltiple de secuencias:
    1. WSClustalW, realiza el alineamiento múltiple de las secuencias introducidas por el usuario para identificar los bloques conservadas en todas ellas.
    2. WSTCoffee, realiza la misma función que WSClustalW.

- Búsqueda de datos:
    1. WSDbfetch, recupera información de diversas bases de datos.
- Predicción de secuencias:
    1. WSInterProScan, Se introduce una secuencia desconocida y sus motivos son comparados con los existentes en sus bases de datos de secuencias para predecir el nombre de la secuencia.

Para poder utilizar estos web services el usuario previamente debe haber descargado en su ordenador un cliente para cada uno de ellos, tanto en Axis como en SOAPLite, y seguir las instrucciones facilitadas para su instalación.

1.3. Análisis de expresión génica

Podemos ver que los campos de estudio cubiertos por la bioinformática son numerosos, pero este proyecto se centra en uno de ellos: el análisis de la expresión génica. Para ello describiremos a continuación como se obtienen los datos para su posterior análisis: la técnica de los microarrays de ADN.

Con esta técnica conseguimos saber como se comportan (datos sobre su expresión génica) miles de genes simultáneamente, todo ello bajo diferentes condiciones iniciales. Es factible pensar que dependiendo de si forma parte de una célula, un tejido o un organismo un conjunto específico de genes se comportará de manera distinta. Para estos casos se hace evidente que el uso de esta técnica en particular es muy valiosa para la obtención de datos sobre como actuarán distintos grupos de genes. También podemos notar que uno de los principales problemas de esta técnica es el gran número de datos que proporciona, por lo que para su tratamiento se debe hacer uso de técnicas como la minería de datos.

1.3.1. Estructura y características del ADN

El material que forma nuestros genes es el ácido desoxirribonucleico (ADN). Es una macromolécula formada por dos cadenas enrolladas en forma de hélice (ver figura 1.2), cuyos componentes son cuatro bases llamados nucleótidos. Cada uno de estos nucleótidos esta formada por un grupo fosfato, una base nitrogenada y una desoxirribosa. Estas cuatro bases se dividen en dos grupos: purínicas, llamadas Adedina (simbolizada por la letra A) y Guanina (G) , y pirimidínicas, llamadas Citosina (C) y Timina (T).

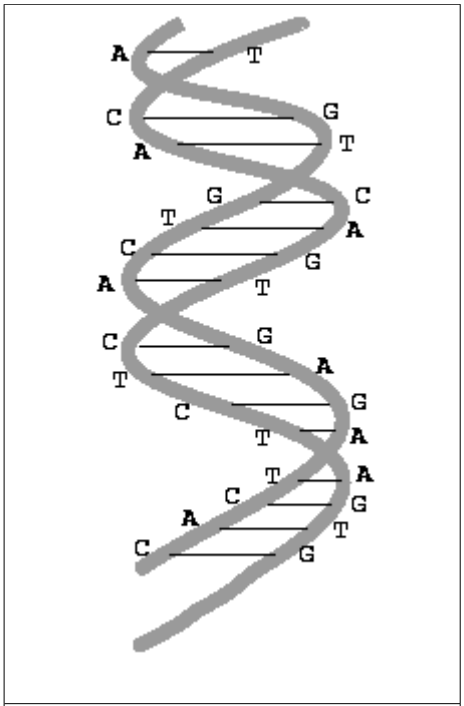

Fig. 1.2 Representación esquemática de la doble hélice de ADN.

La unión de ambas cadenas es debido a que usualmente la adenina se une con la timina mediante dos puentes de hidrógeno y la guanina con la citosina mediante tres puentes de hidrógeno.

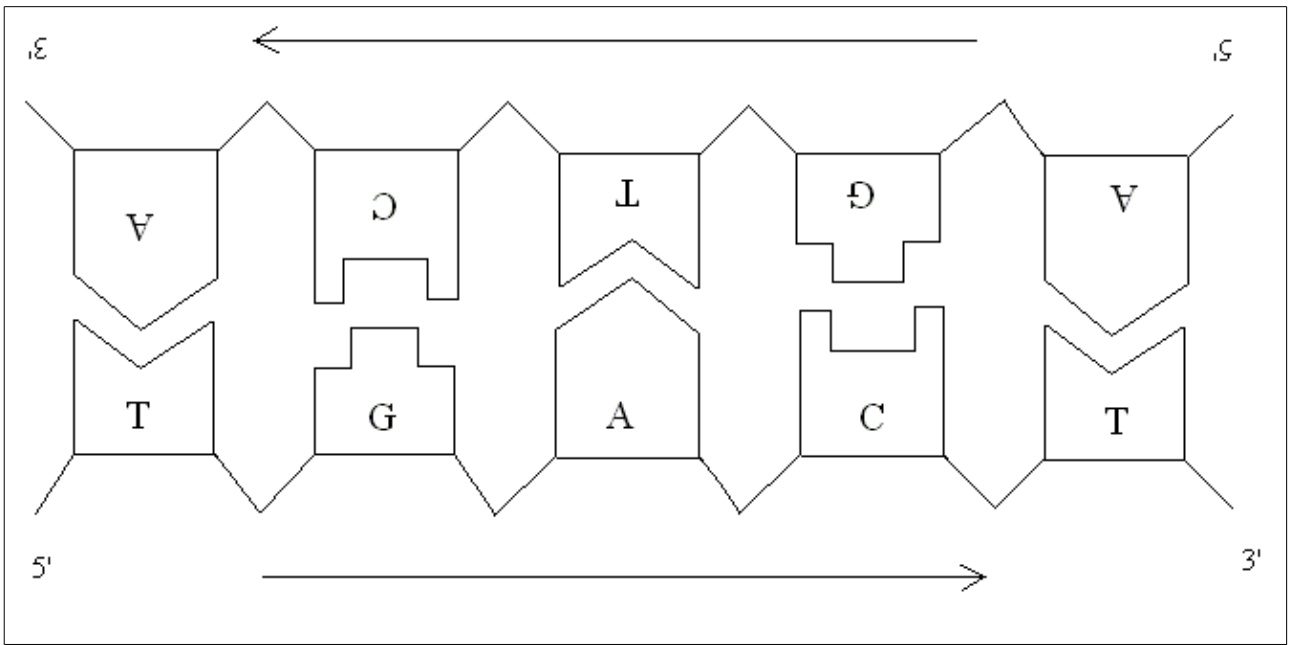

Fig. 1.3 Esquema de la distribución de las cadenas de ADN, donde se aprecia la complementariedad de las cadenas.

Los genes son aquellos fragmentos de ADN que codifican una proteína. Este proceso de traducción del ADN codificado en el gen en una proteína ocurre fuera del núcleo. Por tanto para realizar esta tarea se usa una molécula que es la copia de las cadenas de ADN que se quieren transmitir, llamada ARN mensajero. El Ácido Ribonucleico (ARN) también esta formado por cuatro bases, que a su vez están divididas en dos grupos: las purínicas que son Adenina (A) y Guanina (G), y las pirimidínicas que son Citosina (C) y Uracilo (U).

1.3.2. Técnicas de microarrays de ADN

Básicamente consiste en una matriz en dos dimensiones de secuencias de ADN sobre un sustrato sólido. Estas secuencias son cadenas simples llamadas oligonucleótidos, se compara con cDNA (ADN complementario) , que se obtiene por la síntesis de ADN mensajero. Estas cadenas de ADN complementarias solo reaccionan (hibridan) con los depósitos complementarios de la matriz. Los puntos de la matriz que se han combinado son detectados ya que han sido tratados con una sustancia fluorescente. Por tanto podemos saber que genes son activos con tan sólo observar cuales han sido excitados mediante la aplicación de un detector confocal que los ilumina.

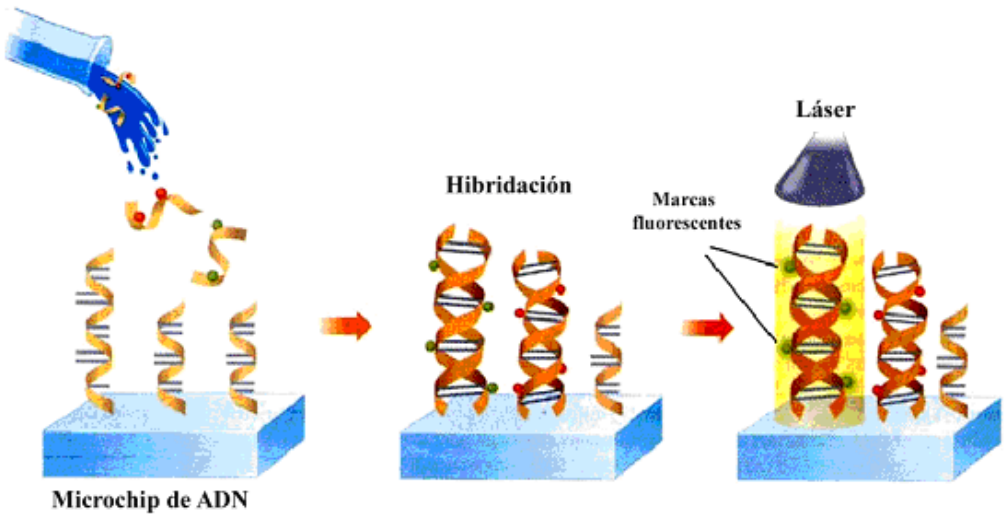

Fig. 1.4 Esquema de la técnica de microarray de ADN. Los ADN complementarios marcados con las sustancias fluorescentes son depositados en el microarray de ADN. En el microarray tenemos dispuestas en la matriz las distintas secuencias de ADN conocidas. Por el proceso conocido como hibridación las secuencias de ADN complementario se unen a las que se encontraban dispuestas en el microarray. Una vez unidas son tratadas con distintos láseres que excitan las sustancias fluorescentes y permiten conocer los lugares de unión de las cadenas.

Podemos concluir que una de las principales funciones de estas técnicas es la detección de los niveles de expresión de ARN mensajero que interviene en el proceso de trascripción de las células.

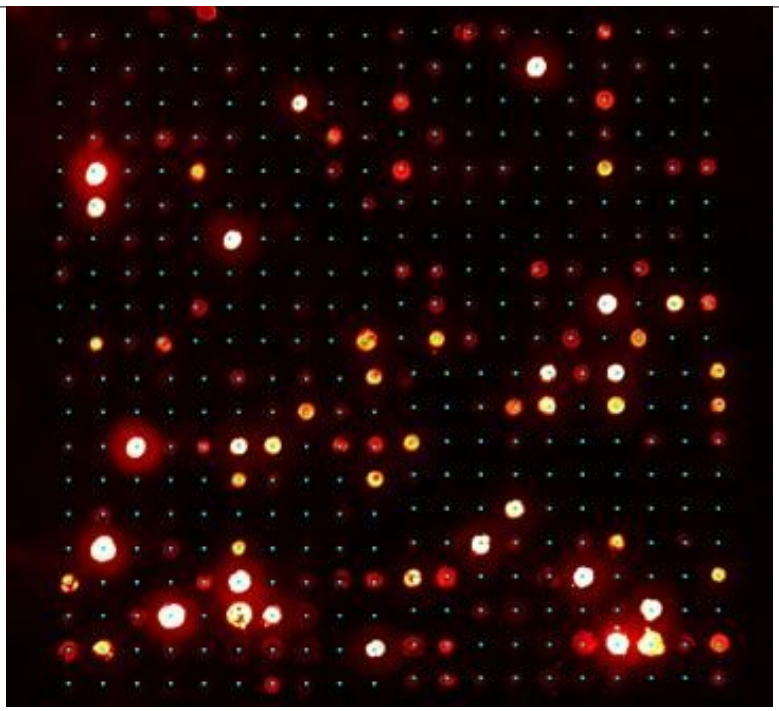

Fig. 1.5. Ejemplo de imagen obtenida mediante la técnica de microarrays [6]. se puede observar la distinta coloración, roja y amarilla, de las expresiones de los genes.

1.3.3. Análisis de los datos

Inicialmente se parte de imágenes monocromas que deben ser tratadas para convertirlas en matrices de valores de expresión. Este procedimiento se realiza utilizando las técnicas clásicas de procesamiento de imágenes. De manera muy básica lo que se hace es obtener una matriz de expresión normalizada en función de la intensidad de los diferentes puntos que conforman la imagen, y su comparación con la intensidad del fondo. De este modo se consiguen suprimir los posibles errores provenientes de los aparatos de medición.

La matriz obtenida generalmente representa en sus filas genes y en las columnas las condiciones experimentales a las que se ven expuestos.

La matriz de expresión puede obtenerse a través un análisis de los niveles de expresión de los genes, en el cual cada gen es un vector y cada punto del vector es una muestra (ver Fig 1.5).

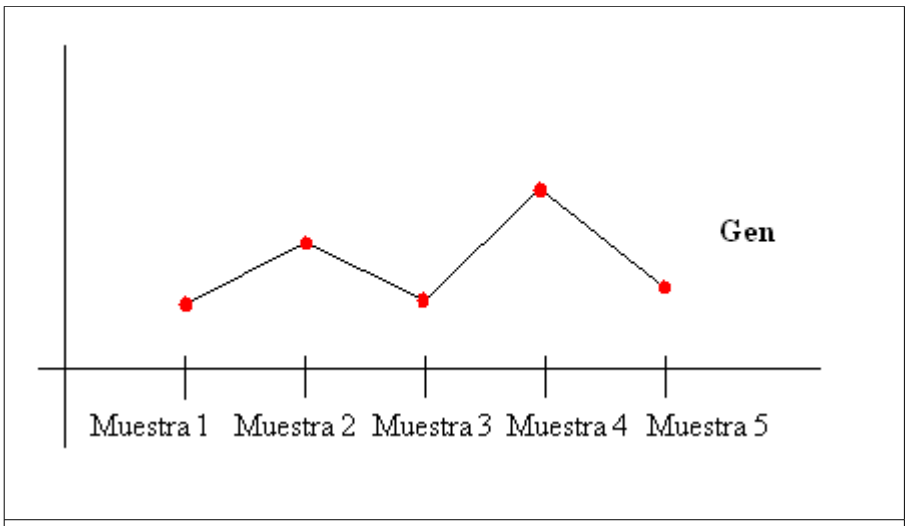

Fig 1.5 Análisis de los niveles de expresión de los genes

Otra técnica es mediante un análisis fenotípico, en el cual cada muestra es un vector y los puntos que los forman son los genes (ver Fig 1.6).

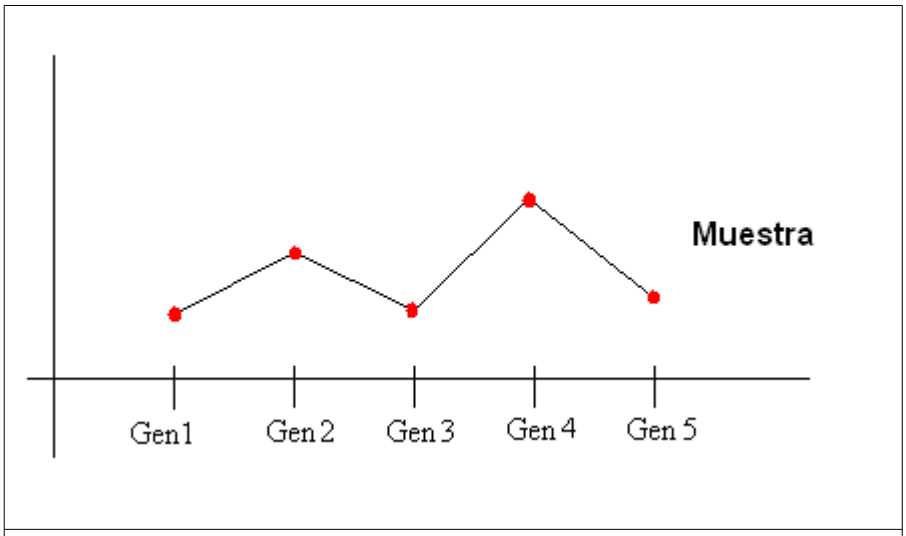

Fig 1.6 Análisis fenotípico

Una vez obtenidos estos datos son procesados mediante métodos de agrupamiento y de este modo tener diferenciados los genes o muestras que presenten propiedades de expresión similares. Antes de utilizar los métodos de agrupamiento los datos o muestras deben ser tratados con métodos de procesamiento, siendo algunos de ellos la transformación logarítmica o la normalización.

2. Una herramienta de análisis de datos de expresión génica: El Paquete Eng*e*ne.

2.1. Motivación

Hoy día está apareciendo una gran cantidad y variedad de datos de expresión génica, los cuales hay que procesar por medio de programas informáticos ya que sería imposible su de forma manual. La bioinformática es la ciencia que se encarga de ello, informáticos junto con biólogos se encargan de desarrollar los programas para el procesamiento de dichos datos.

Actualmente existen diferentes servicios a través de web para el tratamiento de dichos datos tales como el EBI (European Bioinformatic institute), que aparte del procesamiento de datos también gestiona las diferentes bases de datos biológicas de Europa.

Eng*e*ne es una plataforma independiente que se compone de diferentes herramientas para la exploración génica, visualización y procesamiento de grandes conjuntos de datos genéticos. Dichas herramientas realizan agrupamiento de datos por algoritmos particionales o algoritmos jerárquicos.

Eng*e*ne es una herramienta gratuita y sin ningún ánimo de lucro a disposición del mundo científico relacionado con la biología molecular. Los programas que se han utilizado como referencia para desarrollar el *E*ng*e*ne están disponibles en la web [10] del CNB, el *E*ng*e*ne está disponible la siguiente web [8] alojada en un servidor de la UCM o en la web [9] alojada en un servidor de la UAM.

2.2. Función

El paquete Eng*e*ne es un conjunto de programas en C++ y una interfaz web para el tratamiento de datos de expresión génica que se describen a continuación. Se dividen en cinco subconjutos: Preprocesamiento, métodos de agrupamiento, métodos proyectivos, análisis estadístico y reglas asociativas.

1. Preprocesamiento.

Este apartado solo lo compone un programa que realiza tareas de preparación de los datos para su posterior procesado, en la tabla se dan mas datos sobre el mismo.

Utilidad	Programa	Descripción
Preprocesamiento	preprocess	Realiza un preprocesamiento del archivo original de datos ".dat" en el que se ajustan parámetros tales como el filtrado, relleno, normalización, umbrales y centrado. También puede realizar una transposición de la matriz inicial de datos, dicha transposición puede ser necesaria para la interpretación posterior de los datos, ya que depende en gran medida del formato de estos. Tiene como resultado un archivo preparado para ser procesado por cualquier algoritmo de los disponibles en el paquete Eng*e*ne.

2. Métodos de agrupamiento.

El Eng*e*ne consta de seis métodos de agrupamiento, estos métodos consisten en procesar los datos para su posterior agrupamiento dependiendo de las características y similitud de estos. Con este agrupamiento se consigue que sea más fácil el estudio de estos ya que se organizan entorno a distintos conjuntos cada uno con características similares.

En la tabla de a continuación se detalla el tipo de algoritmo o utilidad y el programa que lo realiza.

Utilidad	Programa	Descripción
Agrupamiento jerárquico	hierarchical	Permite agrupar los datos a través de diferentes métodos de agrupamiento jerárquico.
K-means	kmeans	Es un algoritmo de agrupamiento particional, esta utilidad al igual que las demás nos permite establecer parámetros de ejecución tales como el tipo de distancia, normalización o incluso iteraciones.
Fuzzy C Means	fcmeans	Es una utilidad que nos permite agrupar datos en conjuntos o clases difusas. Esta agrupación no se hace en conjuntos completamente separados, los elementos que poseen casa uno pueden tener relación con los de otro conjunto, por eso lo denominamos agrupación difusa.
Kernel C Means	kcmeans	Es un método de agrupamiento particional. El algoritmo agrupa los datos en distintos conjuntos dependiendo de las características de estos. Se puede consultar mas información en la referencia [7].

Utilidad	Programa	Descripción
Fuzzy kohonen clustering network	fkcn	El programa es un algoritmo que combina SOM y métodos difusos produciendo muy buenos resultados.
Double Threshold	dblthr	Este procedimiento pone juntos los datos cuya distancia esta bajo la especificada por el parámetro umbral, y separa los datos si la distancia supera la especificada en el umbral.

3. Métodos de proyección

Existen otros tipos de métodos que trata los datos de forma diferente que en el apartados anterior, esto son los métodos proyectivos. El objetivo principal de estos métodos es proyectar los datos de entrada en un espacio bidimensional para su posterior tratamiento, los datos de entrada pueden llegar en espacios de mas de dos dimensiones en los que a veces es difícil su lectura.

En la siguiente tabla se especifican las seis utilidades del Eng*e*ne capaz de realizar estos trabajos.

Utilidad	Programa	Descripción
Componentes principales de análisis	pca	Transforma el sistema de coordenadas en el cual están representados los datos. Los nuevos valores de coordenadas mediante los cuales se representan los datos le llamamos componentes principales, con esto se logra que sea más fácil el estudio de dichos datos.
Sammon	sammon	Es una técnica de mapeado de datos no lineal en la que se intenta conseguir pasar datos de entrada que están en varias dimensiones a solo dos dimensiones preservando distancias y relaciones.
SOM	som	Este algoritmo implementa los mapas auto-organizativos de Kohonen, que obtienen en muchos menos conjuntos los datos mas representantes de los originales. Se puede consultar mas información en [11].
Batch SOM	batchsom	Implementa los mapas auto-organizativos de Kohonen usando la variante “Batch training”. Se puede encontrar mas información respecto a esa variante en [6].

Utilidad	Programa	Descripción
Fuzzy SOM	fsom	Este algoritmo deriva del SOM, utiliza técnicas difusas para el tratamiento de los datos. Puede servir como referencia la información [12].
KerDenSOM	kerdensom	"Kernel Probability Density Estimator Self-Organizing Map", así se define el algoritmo que ejecuta este programa y también es una variedad de los anteriores. Se puede consular más información en [7].

4. Análisis estadístico.

Las utilidades que se describen a continuación simplemente obtienes información estadística de los datos de entrada.

Utilidad	Programa	Descripción
Histograma de distacias	statsig	Obtiene los datos de la distribución de distancias.
Histograma de valores	valhis	Obtiene los datos de la distribución de valores.

5. Reglas asociativas.

En el Eng*e*ne existe únicamente un programa que realiza dicha acción, el “tranextr” que se especifica en la siguiente tabla.

Utilidad	Programa	Descripción
Transaction extraction	tranextr	Produce un conjunto de transacciones para finalmente poder aplicar una serie de reglas asociativas a los datos de entrada.

2.3. Arquitectura

Engene funciona gracias ha unos conjuntos de programas y librerías realizadas en C++ para el tratamiento de datos de expresión génica.

En los programas que lo componen existen dos grandes grupos, el primero y más importante es el grupo de programas de procesamiento y agrupamiento de datos, y el segundo grupo es para la visualización en imágenes de dichos datos.

La estructura de directorios es la siguiente:

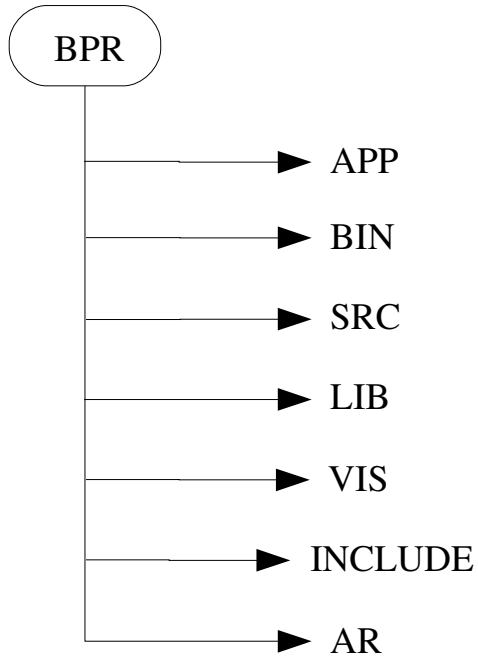

Fig. 2.1. Estructura de directorios en Eng*e*ne

- APP: Código fuente de las aplicaciones para el procesamiento de datos.
- BIN: Binarios ejecutables de todas las aplicaciones.
- SRC: Código fuente de las librerías dinámicas.
- LIB: Librería dinámica.
- VIS: Código fuente de los programas de visualización.
- INCLUDE: Librerías estáticas.
- AR: Códigos fuente de los programas de reglas asertivas.

El Eng*e*ne no tiene ningún tipo de base de datos para el procesamiento de la información todo los trabajos los realiza a partir de los algoritmos vistos anteriormente, a diferencia de otros servicios webs que lo que hacen son comparaciones de datos con bases de datos públicas.

Inicialmente el Eng*e*ne esta montado en una web [10] escrita en el lenguaje PHP, dicha web permite procesar archivos de datos genéticos y mostrar los resultados. El funcionamiento es sencillo, la web o front-end llama a los programas correspondientes para el tratamiento de los datos. Incluso nos permite la visualización interactiva con la aplicación JAVA que lleva incluida.

2.4. Ejemplo de funcionamiento a través de la web

En esté ejemplo se describe el procesamiento de un conjunto de datos con el algoritmo SOM (ver sección 2.2.3). A través de su web [10] se puede hacer dicho ejemplo.

El primer paso de todo es registrarse en la web a través del formulario correspondiente.

Una vez que se ha accedido aparece un directorio donde se puede subir los archivos que se deseen, se busca el archivo pinchando en el botón “examinar” y una vez seleccionado pinchamos en “upload”, cuando el archivo este en el servidor nos aparecerá dentro de la carpeta (Figura 2.2.).

Figura 2.2.

Primero debemos preprocesar el archivo, para ello pinchamos en el archivo y se mostrará una pantalla con datos del archivo (Figura 2.3.).

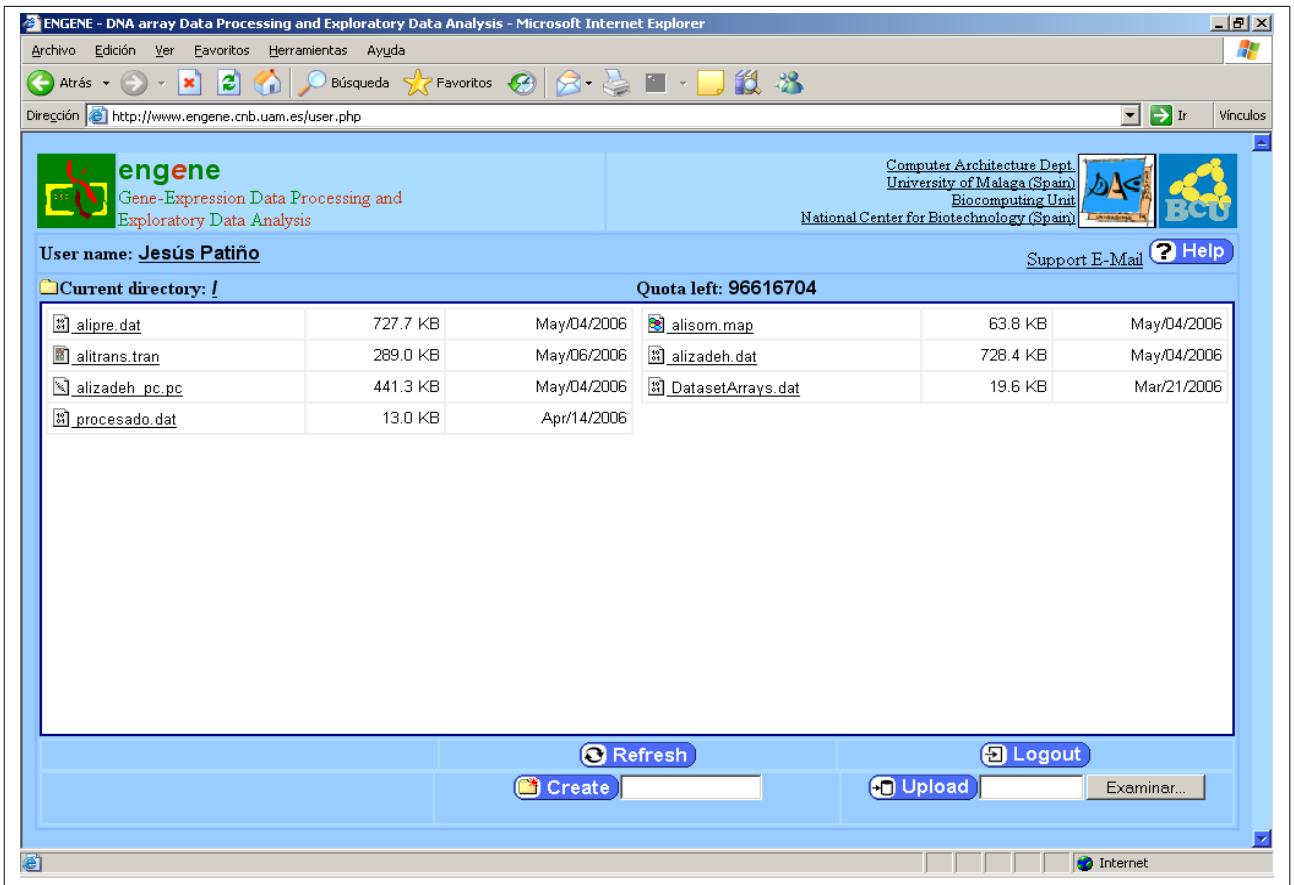

Fig. 2.2. Directorio Engene

Figura 2.3.

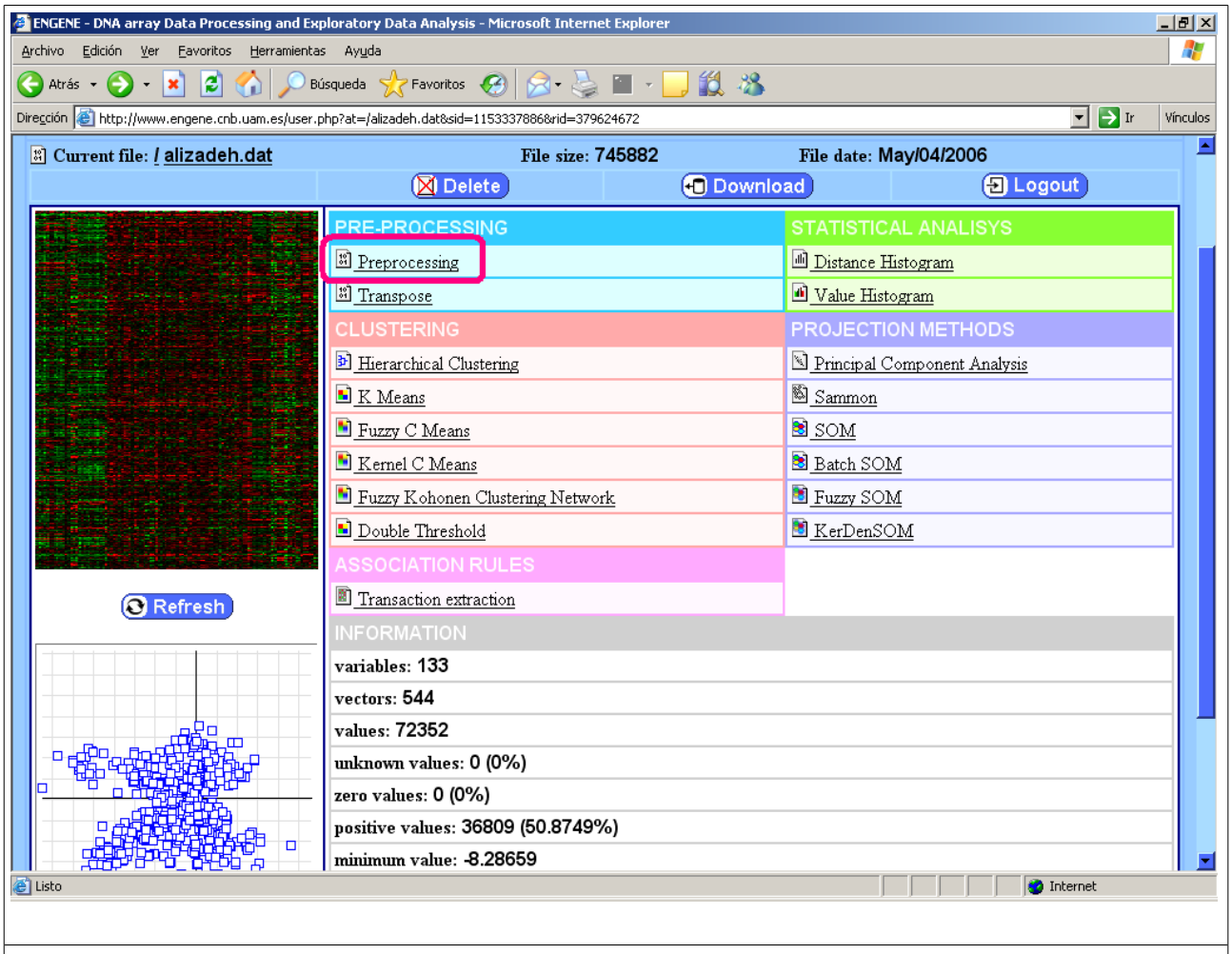

Fig. 2.3. Se elige Preprocessing

En dicha pantalla se accede a “Preprocessing” y rellenamos los campos correspondientes, en este caso el único obligatorio es el nombre del archivo de salida, los demás parámetros en principio toman valores por defecto (Figura 2.4.).

Figura 2.4.

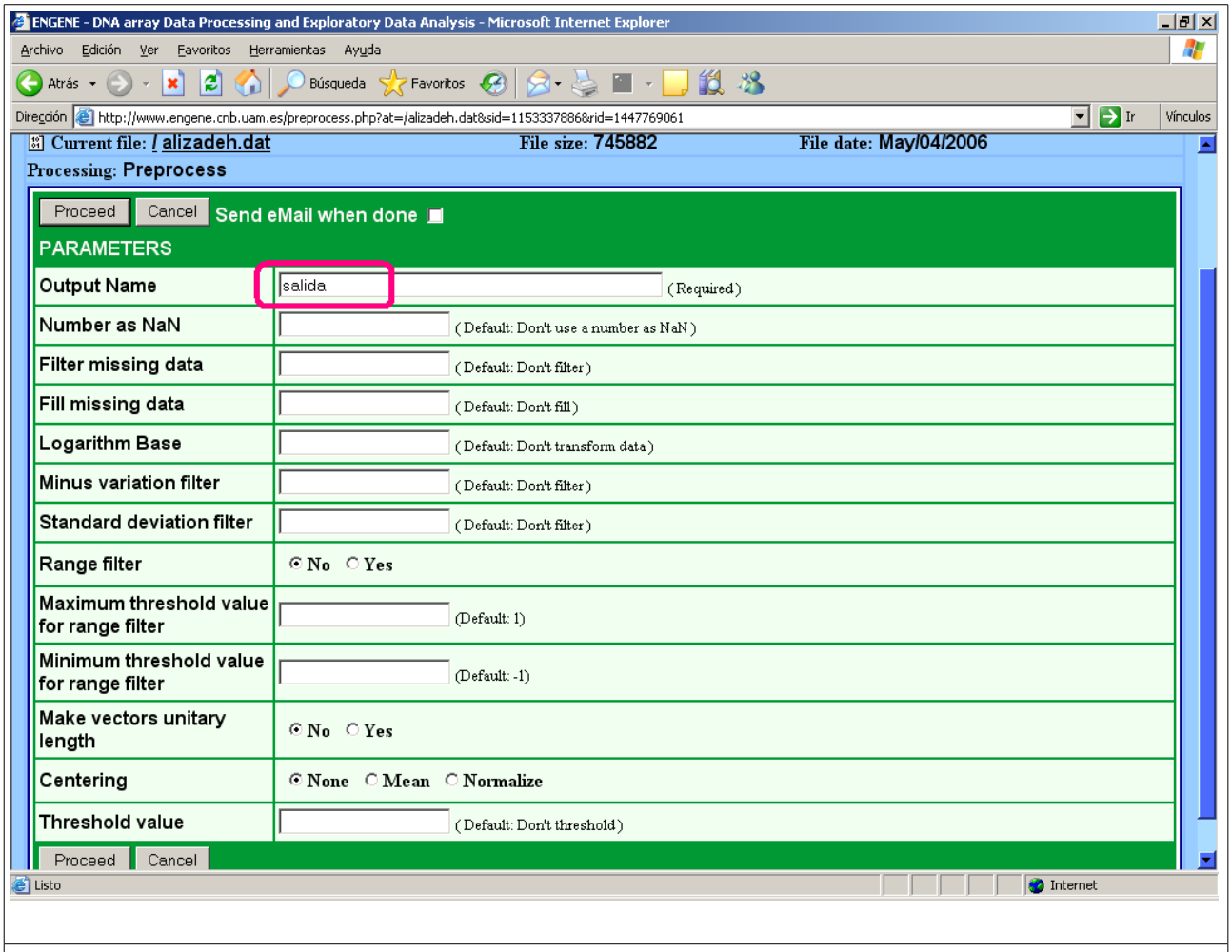

Fig. 2.4. Descripción de la entrada a Preprocessing, donde se indica que es necesaria la entrada de un archivo de descripción de datos.

Una vez preprocesado el archivo observamos que aparece en el directorio de ficheros que tenemos el archivo de salida con extensión ".dat", pinchando en dicho archivo aparecerá información del mismo y la opción de ejecutar el algoritmo de SOM (Figura 2.5.).

Figura 2.5.

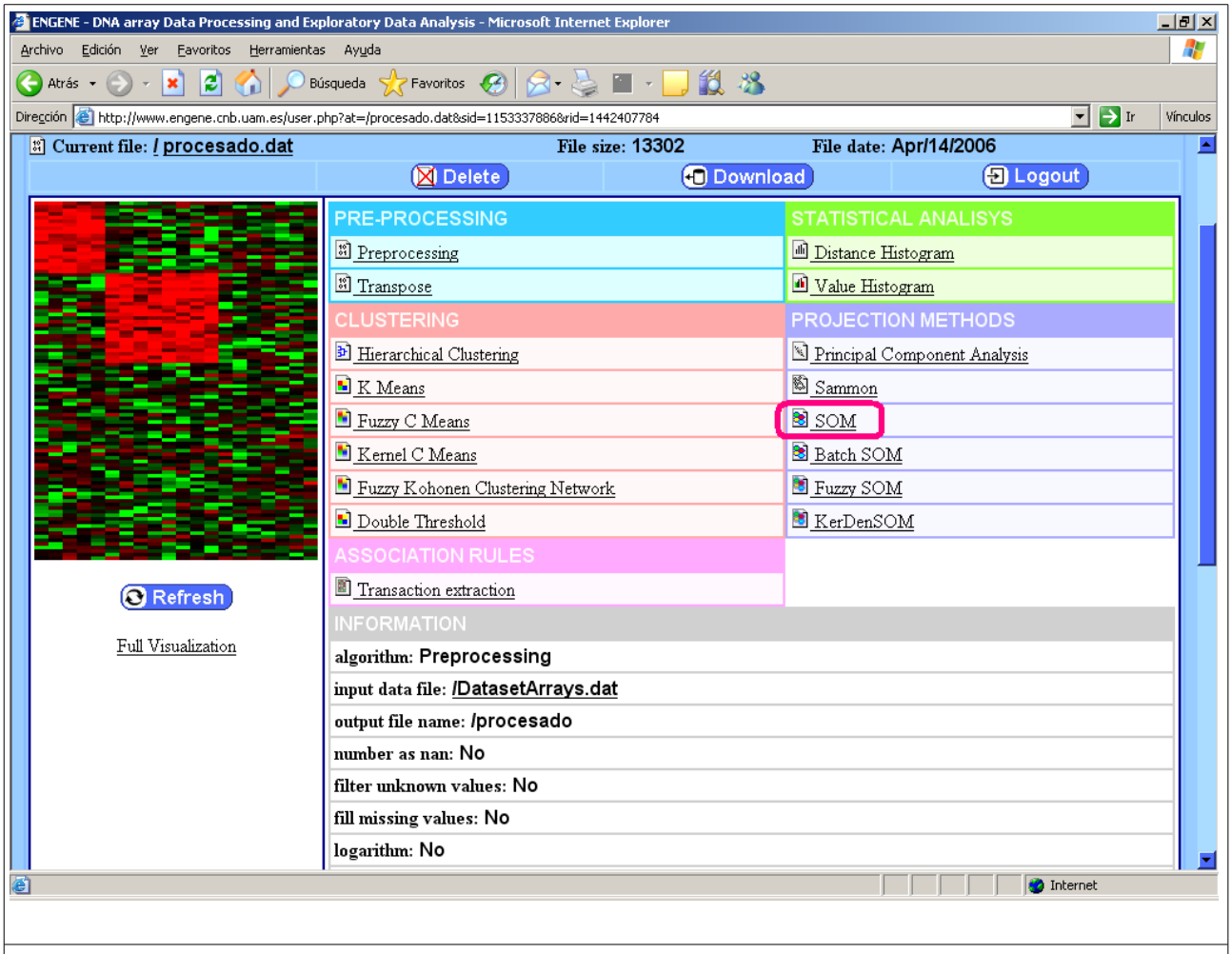

Fig. 2.5. Se elige Som

El intentar ejecutar este algoritmo también pedirá un conjunto de parámetros, en principio para este ejemplo se dejan todos por defecto menos el nombre de salida del archivo, tal como se muestra en la figura 2.6.

Figura 2.6.

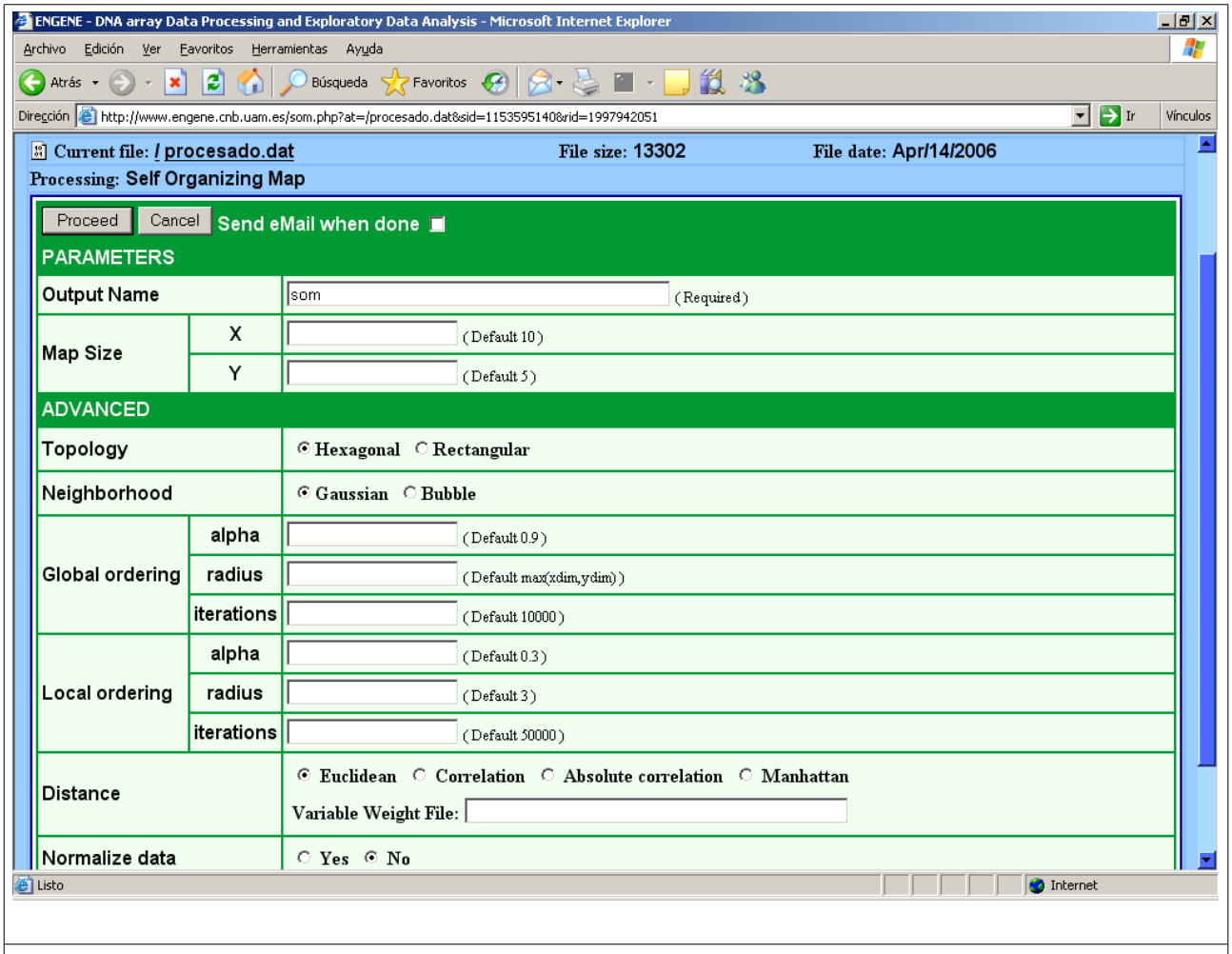

Fig. 2.6. Conjunto de archivos y parámetros necesarios para el algoritmo Som

Una vez que se pincha en el botón “Proceed” se ejecuta el algoritmo y se genera un archivo con extensión “.map”, en el ejemplo este se llamaría “som.map”. Accediendo a los datos de este fichero se puede ver el resultado de ejecutar el algoritmo (Figura 2.7.).

Figura 2.7.

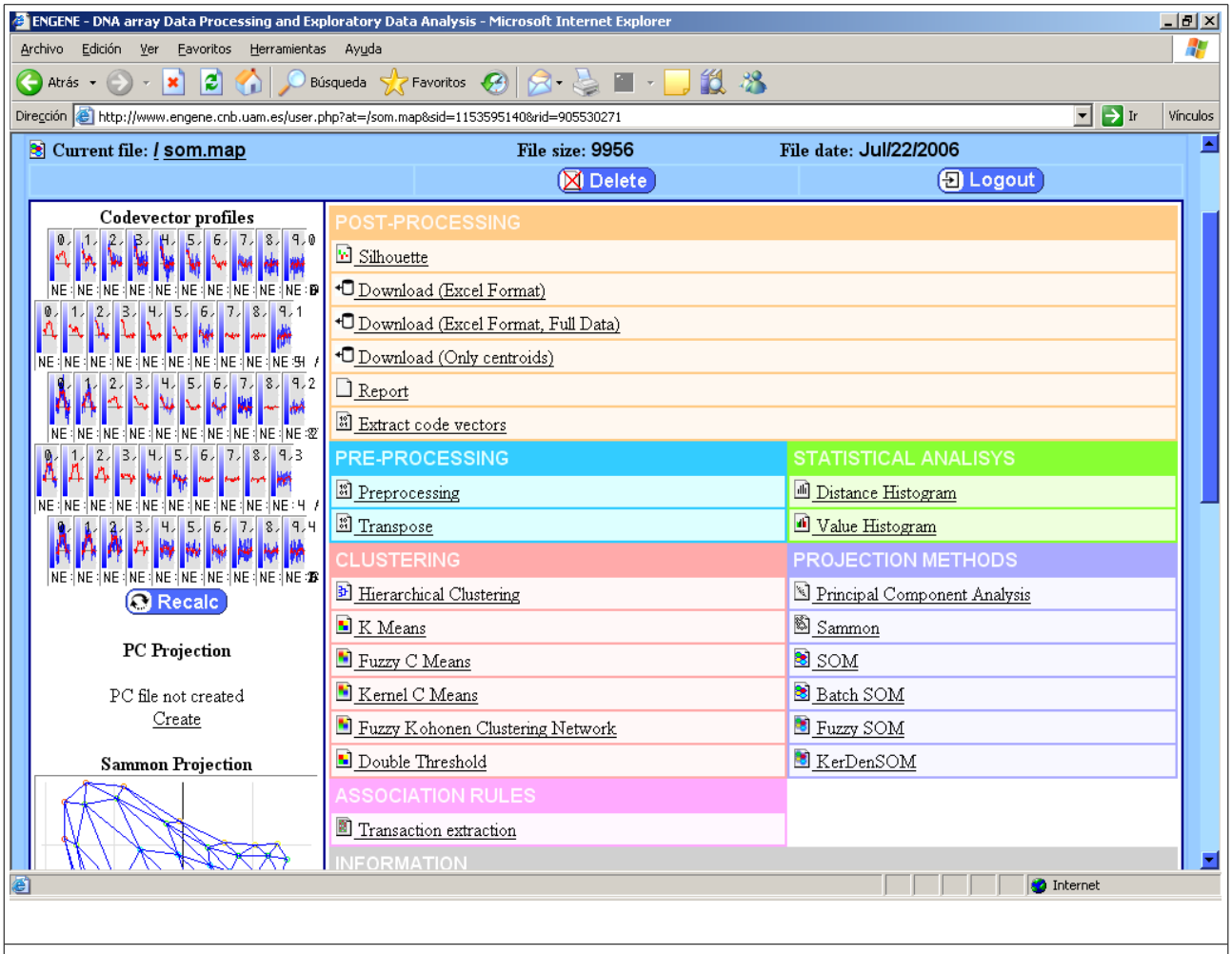

Fig. 2.7. Resultados de la ejecución del algoritmo Som

Como se puede observar estos datos van acompañados de imágenes que como se ha comentado anteriormente el propio paquete Engene tiene programas para su generación (Carpeta vis).

3. Objetivos del proyecto

Las actuales aplicaciones informáticas que se pueden encontrar en la actualidad presentan un problema, y es que no permiten el intercambio de información entre ellas. Este conlleva una falta de cooperación entre los distintos grupos de investigación. Con la creación de un servicio web (web services) que permita ofrecer una capa de abstracción mayor que la actual se conseguirían dos objetivos. El primero de ellos ocultar detalles de la implementación y ofrecer una aplicación trasparente y sencilla de utilizar. Y el segundo obtener una solución técnica al problema de integración de los diferentes métodos de análisis.

Por ello se pretende crear una plataforma de análisis de datos basada en servicios web, y de este modo disponer de una herramienta que permita la interoperabilidad entre distintas aplicaciones bioinformáticas. Para cumplir este objetivo se ha hecho uso de los métodos contenidos en el paquete Eng*e*ne, el cual contiene una colección de programas enfocados al análisis y exploración de datos de expresión génica. Este software es de dominio público y es accesible en la siguiente dirección: http://www.engene.cnb.uam.es

Si bien todas las aplicaciones disponibles para realizar el análisis de datos son útiles, tienen el inconveniente de que se hace necesario que sea un usuario humano el que solicite la información a través de la web. Esto no es necesario gracias a los servicios web, ya que no son los usuarios los que solicitan la información directamente, si no que lo realizan aplicaciones sin la intervención de estos.

4. Descripción de servicios web

4.1. Introducción

El W3C define un servicio web como un sistema software diseñado para soportar comunicación máquina-máquina de forma interoperable en una red.
Un web service permite a dos aplicaciones remotas conectarse y enviarse datos. Cada servicio web tiene un protocolo específico, que está definido de forma independiente de arquitectura/sistema operativo o lenguaje de programación. Para conseguir interoperabilidad en el envío de datos, se utiliza XML como envoltorio de los mismos. [15]

Este modelo del uso de la web podría verse como una sustitución del escenario típico de utilización de la world wide web, en el que una persona utiliza un navegador para enviar peticiones a un servidor remoto. El usuario es el actor principal, el que inicia las peticiones web [14]

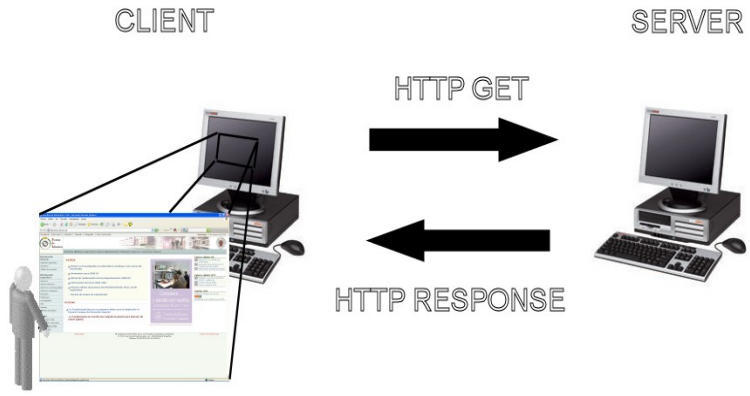

Fig. 4.1 Modelo del uso de la web

En el modelo de los servicios web, que es "centrado en aplicación". Son las aplicaciones las que se conectan directamente. El usuario ya no maneja un navegador sino una aplicación, lo que permite muchas veces, una utilización del servicio más automática y eficiente.

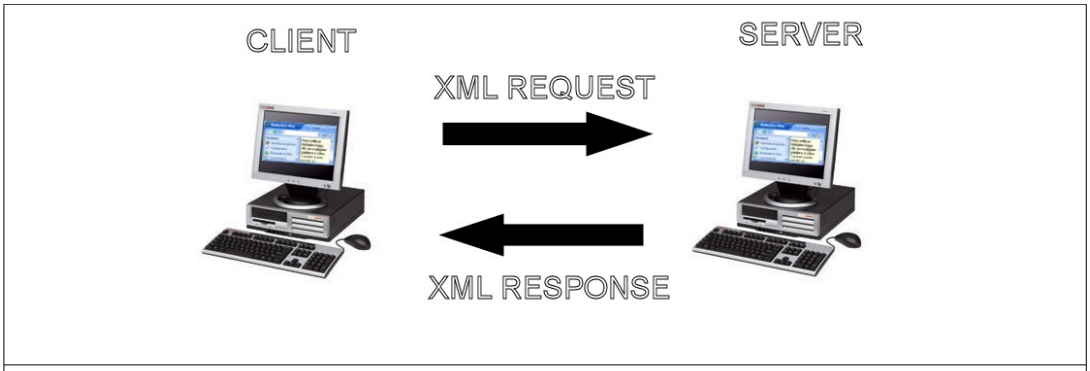

Fig. 4.2 Modelo de los servicios web

Un ejemplo de esta forma de utilizar la web podría ser un servicio de traducción de palabras. Imaginemos que el usuario tiene una palabra escrita en la pantalla y quiere traducirla. Si el servicio se realiza mediante un página web típica, la realización de la traducción pasa por:

1) El usuario debe indicar al navegador la dirección donde debe conectarse.
2) Se descarga una página HTML, con texto e imágenes.
3) El usuario puede entonces escribir la palabra y pulsar un botón para proceder.
4) Se carga una nueva página HTML de forma completa que ha enviado el servidor, donde se indica la traducción.

En un enfoque basado en aplicación para resolver el problema, podría haber en el sistema del usuario un servicio funcionando que permitiese pulsar manteniendo una tecla de control sobre cualquier palabra para traducirla. Con lo que en este enfoque quedaría:

1) El usuario pulsa pincha sobre la palabra manteniendo una tecla pulsada que indica a la aplicación de traducción.
2) La aplicación cliente envía los datos al aplicación servidor.
3) La aplicación servidor responde y aparece en la pantalla del cliente la traducción

El enfoque basado en aplicación, que podría ser implementado con un web service, ha permitido una resolución más automática del problema ya que el usuario no ha tenido que escribir de nuevo la palabra, ni recordar la dirección de la página de traducción. Además se ha eliminado la sobrecarga de información que haya podido introducir la descarga de la presentación en imágenes y texto de la página web de traducción, con lo que se ha hecho un uso más eficiente del ancho de banda de conexión del usuario.

4.2. La pila de protocolos de los servicios web [16]

La pila de protocolos de web services es una colección de protocolos utilizada para definir, localizar, descubrir e implementar servicios web. Está formada por cuatro capas:

1) Capa de transporte: responsable del transporte de mensajes entre aplicaciones en la red. Por ejemplo HTTP, FTP o SMTP.
2) Capa de mensajería XML: responsable de la codificación de mensajes en un formato XML común, de forma que pueda ser comprendida en los dos extremos de la conexión. Por ejemplo SOAP o XML-RPC.
3) Capa de descripción de servicio: especifica el interfaz del servicio web. WSDL es el estándar en esta capa.
4) Capa service discovery: centraliza los servicios en un registro común, de forma que los servicios web pueden publicar su localización y descripción y hace fácil que sean descubiertos. Ejemplo: UDDI.

La corriente principal de uso de los servicios web es utilizar HTTP como capa de transporte, SOAP como capa de mensajería, WSDL para la descripción del servicio y UDDI para el descubrimiento del servicio con lo que serán están tecnologías las que se describirán en apartados siguientes. XML-RPC en realidad evolucionó a lo que es hoy SOAP.

Existen otros protocolos de reciente adopción aplicables también a la pila de servicios web entre los que destacamos:

a) WS-Security: añade características de seguridad al servicio, incorporándolas al paquete SOAP. Entre ellas están la autenticación, el establecimiento de permisos para el host que realiza la llamada al servicio y el añadido de atributos sobre para el host cliente.
b) WS-Reliability: permite a un servicio web cumplir requisitos críticos de fiabilidad que de otra forma no podrían ser logrados con el conjunto formado por SOAP sobre HTTP. Su especificación define un servicio web fiable como aquel en el que cada mensaje llega una vez y sólo una vez al destinatario y en el orden correcto.

Un mecanismo de mención es WS-Adressing que permite a los web services comunicar información de direccionamiento. Se compone esencialmente de dos partes:

1. Endpoint reference, que incluye, entre otras cosas, la dirección donde va dirigida el mensaje.
2. Message Addresing Properties: otra que le asocia propiedades de direccionamiento, como el origen del mensaje, a dónde debe ser dirigida la respuesta o a donde debe ser mandados los mensajes de error.

4.2.1. La capa de mensajería: SOAP [14]

SOAP intercambia mensajes XML sobre una red, normalmente sobre HTTP.
XML fue elegido como formato para la codificación de mensajes por su gran aceptación actual; está siendo utilizado al mismo tiempo por las grandes corporaciones y por los desarrollos open source. Los mensajes XML contienen tags, que son “metadatos”, datos que son información sobre los propios datos. De esta forma los mensajes SOAP son fácilmente entendibles para un ser humano, pero ligeramente costosos de manejar para una máquina. De hecho si los mensajes son largos, SOAP puede llegar a ser 10 veces más lento que otras tecnologías que utilizan codificación binaria como RMI.

La elección típica de utilizar HTTP como capa de transporte, tiene que ver con que de esta forma los servicios web funcionan bien en la infraestructura actual de internet. Concretamente SOAP funciona bien con los firewalls, lo que le da una ventaja sobre otros protocolos que son filtrados por los firewalls como el protocolo DCOM de Microsoft.
Sino se utiliza ningún protocolo especial, como WS-Adressing, los roles cliente/servidor implicados son fijos, sólo el cliente puede utilizar los servicios del servidor.

Ejemplo:

En el servicio de traducción anteriormente descrito, supongamos que existe una operación traducir. Una captura de los paquetes de esa operación completa podría dar como resultado. Como se puede apreciar, en este caso se ha escogido HTTP como capa transporte.

4.2.1.1 .- La petición SOAP

La petición del cliente debe incluir el nombre del método a invocar junto con sus parámetros. Como ejemplo, en el servicio de traducción anteriormente descrito, podríamos tener un parámetro de nombre palabra y valor “suerte”:

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
                   xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
                   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                   xmlns:traduccion="http://www.fdi.ucm.es:8080/traduccion.wsdl">
 <SOAP-ENV:Body SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <traduccion:traducir>
   <palabra>
   suerte
   </palabra>
  </traduccion:traducir>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

En este mensaje se puede observar la inclusión de un elemento Envelope, que a su vez incluye un elemento Body; ambos han de aparecer obligatoriamente en un mensaje SOAP.
También se aprecia la definición de cinco nombres de espacios. Los nombres de espacio permiten desambiguar identificadores propios del servicio al que correspondan el mensaje y los pertenecientes a SOAP. Además permiten a los procesadores SOAP distinguir entre las versiones 1.1 y 1.2 de SOAP, ya que los nombres de espacios entre ambas versiones son diferentes.

4.2.1.2 .- La respuesta SOAP

Una posible respuesta de ese mismo ejemplo de sistema de traducción sería:

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
                   xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
                   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                   xmlns:traduccion="http://www.fdi.ucm.es:8080/traduccion.wsdl">
 <SOAP-ENV:Body SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <traduccion:traducirResponse>
   <palabra_ingles>
   luck
   </palabra_ingles>
  </traduccion:traducirResponse>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

De nuevo se incluyen los elementos obligatorios Envelope y el Body. En el Body se indica a qué operación se está llamando, junto con el parámetro palabra_ingles que es la traducción de la palabra en español enviada en la petición SOAP.

4.2.1.3.- El mensaje SOAP

- Envelope:
  Donde pueden indicarse la versión de SOAP mediante la definición de su nombre de espacios,
  SOAP 1.1
  http://schemas.xmlsoap.org/soap/envelope
  SOAP 1.2
  http://www.w3.org/2003/05/soap-envelope

- Header:
  No es obligatorio y su contenido no está especificado por el estándar sino que queda libre para los implementadores de un servicio. Típicamente incluye información de autenticación o datos para el manejo de la sesión. No obstante, el protocolo incluye dos atributos:
  1. Actor:
     En el protocolo SOAP se define un message path como un lista de nodos de servicios SOAP. Cada uno de estos nodos puede hacer algún procesamiento y luego mandar el mensaje hacia el siguiente nodo. Mediante el atributo actor puede especificarse el destinatario de la cabecera SOAP.

  2. MustUnderstand:
     Indica si el elemento Header es opcional u obligatorio. Cuando es true debe

entender la cabecera y por lo tanto procesarla, o devolver un fallo.

- Body:

  En el cuerpo de un mensaje SOAP se incluye la lógica de la aplicación; las peticiones y respuestas.

  Fault
  Es un elemento del Body que se incluye en el mensaje en caso de error. Puede incluir:
  faultCode: un código en texto para indicar una clase de error.
  faultString: una descripción textual del error
  faultActor: una cadena de texto indicando quien causó el fallo, útil en los message paths
  detail: un elemento usado para llevar errores específicos de la aplicación

4.2.2.- Codificación SOAP

La especificación original SOAP se creó antes de que existiesen reglas estándar para la codificación de datos en XML. La especificación actual utiliza XML Schema para la codificar los datos. Como existen algunos tipos de datos no estandarizados por XML Schema, se mantienen algunas convenciones procedentes de la especificación de tipos original como las de arrays y referencias.
En general estas reglas de codificación quedan ocultas por el toolkit SOAP utilizado para el programador de web services.

```
</message>
<message name="traducirResponse">
<part name="palabra_ingles" type="xsd:string"/>
</message>
```

En nuestro ejemplo, la petición incluye un elemento part (un parámetro que llevará la petición) llamado palabra, la palabra en español a traducir. El atributo type debe ser especificado como un XML Schema Qname: el valor del atributo debe estar calificado con el nombre de espacio de XML Schema obligatoriamente. En este caso el tipo es el xsd:string. La respuesta tiene el mismo formato, con la traducción de la palabra al ingles. Múltiples parts pueden ser especificados para cada mensaje.

portType:

En este elemento se definen qué mensajes componen cada operación del servicio.

```
<portType name="TraduccionPortType">
<operation name="traducir">
<input message="traduccion:traducirRequest"/>
<output message="traduccion:traducirResponse"/>
</operation>
</portType>
```

De la misma forma que el atributo type de los elementos part, el atributo message debe estar especificado por un Qname.

WSDL soporta 4 patrones de operación:

- one-way

El servicio recibe un mensaje, pero el cliente no envía una respuesta

- request-response

  El servicio recibe un mensaje y responde con otro. Este el patrón que utiliza la operación traducir de nuestro ejemplo. Un fault element podría ser también incluido en la respuesta.

- solicit-response

  En este caso es el servicio quien inicia la comunicación y el cliente es quien responde. Un fault element podría ser devuelto al servicio

- notification

  El servicio envía un mensaje al cliente y este no envía respuesta

Binding:

Este elemento especifica cómo serán transmitidas las operaciones por la red, es decir, cómo se llevará a cabo el elemento portType previamente descrito:

```
<binding name="traduccion_binding" type="traduccion:TraduccionPortType">
 <SOAP:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
<operation name="traducir">
<SOAP:operation soapAction=""/>
<input name="traducirPeticion">
<SOAP:body namespace="http://www.fdi.ucm.es:8080/traduccion.wsdl" use="encoded"/>
</input>
<output name = "traducirRespuesta">
<SOAP:body namespace="http://www.fdi.ucm.es:8080/traduccion.wsdl" use="encoded"/>
</output>
</operation>
</binding>
```

En este caso, el tag <binding name="traduccion_binding" type="traduccion:TraduccionPortType"> está diciendo “voy a dar los detalles específicos de cómo la operación traducción va a ser llevado a cabo”.
Aunque WSDL es un lenguaje de descripción de servicios, que no tienen porqué estar escritos en SOAP, incluye extensiones específicas para SOAP. Entre estas extensiones se encuentra:

- soap:binding

Que indica que las operaciones se transmitirán con SOAP.
Un estilo “rpc” indica que los mensajes de las peticiones llevarán un wrapper XML indicando el nombre de la función, y a su vez los parámetros de la petición iran dentro de ese wrapper (ver listado 1). Por contra, un estilo document significa la ausencia de un formato predeterminado, tan sólo tiene que ser XML. Cliente y servidor deben de llegar de acuerdo de alguna forma para diferenciar a qué operación se está llamando y con qué parámetros. Por ejemplo el toolkit gSOAP 2.7.6d, en estilo document, intenta averiguar la operación a la que se está llamando por los tipos de los parts enviados. Si dos tipos de peticiones comparten los mismos tipos de parámetros, entonces el mensaje de petición podría llegar al handler de la operación equivocada.

- El atributo “transport” indica que capa de transporte irá debajo de los mensajes SOAP. http://schemas.xmlsoap.org/soap/http indica que será HTTP, mientras que http://schemas.xmlsoap.org/soap/smtp indica un transporte SOAP SMTP

- soap:operation

Proporciona información sobre la operación en conjunto. Su atributo soapAction es opcional en SOAP 1.2 y es una URI que indica el objetivo de la petición. Si se especifica, se deberá indicar un parámetro en la cabecera HTTP con nombre soapAction y valor el especificado. De esa forma la petición podría ser rápidamente despachada, ya que no sería necesario examinar el propio mensaje SOAP para hacerla llegar a su destino, sino que bastaría examinar el paquete HTTP que lo envuelve. También puede ser tenido en cuenta por los firewalls en la decisión de bloquear o no un paquete. Como en nuestro ejemplo se especifica un soapAction de valor la cadena vacía, la petición de nuestro ejemplo de traducción tiene la siguiente cabecera HTTP:

```
POST / HTTP/1.1
Host: localhost:8080
User-Agent: gSOAP/2.7
Content-Type: text/xml; charset=utf-8
Content-Length: 523
Connection: close
SOAPAction: ""
```

- El atributo “use” especifica la codificación que tiene el mensaje SOAP en el sentido de cómo se serializan/deserializan los datos. Puede tener dos valores:
  - literal: indica que las definiciones de tipos siguen un XML Schema.
  - encoded: indica que las definiciones de tipos son en XML, pero no siguen un XML Schema, sino la estructura indicada en una URI especificada por el valor del atributo encodingStyle. Esta opción se utiliza cuando no se quiere que los mensajes no sigan una estructura en árbol, con un elemento raíz, en el cuerpo del mensaje.
- El estilo de binding junto con las reglas de codificación dan un par, style/use, que se utiliza para identificar el modelo de uso que debe aplicarse al servicio. Así se generan cuatro posibilidades:
  - RPC/encoded
  - RPC/literal
  - document/encoded
  - document/literal

De esos cuatro, los más utilizados son document/literal y RPC/encoded. Que el estilo sea RPC no tiene nada que ver con qué se use para modelos de programación RPC. Tan solo indica como traducir un WSDL binding en un formato para el mensaje SOAP.

- Service

Define los ports soportados por el servicio. Cada protocolo soportado en el servicio tiene un correspodiente elemento port. Cada port tiene una dirección del servicio y un “binding” asociado. El elemento service también puede incluir un elemento documentation para describir el servicio:

```
<service name="traduccion">
<documentation> Servicio de traducción inglés-español </documentation>
<port name="traduccion_port" binding="traduccion:traduccion_binding">
<SOAP:address location="http://localhost:8080"/>
</port>
</service>
```

En este caso sólo hay un protocolo, de binding traduccion_binding.

5. Descripción de los servicios web desarrollados

5.1. Introducción

La principal característica del web service que se quería desarrollar era que debía ser capaz de transportar gran cantidad de datos de forma fiable. Los datos provenían de ficheros, la entrada y la salida de los programas de línea de comandos del paquete engene. Por tanto una posible implementación del servicio enviaría los datos como archivos adjuntos a los mensajes SOAP.
En el momento de la construcción del servicio, los web services manejaban principalmente las tecnologías MIME (Multipurpose Internet Mail Extensions) y DIME (Direct Internet Message Encapsulation) para los attachments. Ninguna de las dos posibilidades son recomendaciones del W3C, es decir no están reconocidas por él como un estándar terminado, son sólo propuestas. No obstante, los toolkits de generación de código más famosos como Axis o SOAP:Lite las soportan. Un mensaje enviado por MIME suele ser mantenido enteramente en memoria antes de poder ser volcado a un dispositivo secundario, como el disco duro. DIME sin embargo envía los datos en streaming, con lo que los datos pueden ser volcados a medida que van llegando en pequeñas partes. Esto hizo que escogiésemos DIME para el envío de los ficheros, ya que hacía el envío de datos mucho más eficiente.

Un ejemplo de utilización de DIME para el transporte de datos en bioinformática es BIND (Biomolecular Interaction Network Database) que es una base de datos diseñada para almacenar descripciones completas de interacciones, complejos moleculares y rutas metabólicas. BIND ofrece el envío de datos tanto mediante MIME como DIME.[20][21]

5.2. El toolkit de generación de código [17]

Desde un principio se tomó la decisión de desarrollar el servidor del web servicio en C++. Los ejecutables del paquete Eng*e*ne sólo funcionaban en linux por lo que la portabilidad mayor que proporcionaba Java no tendría demasiada utilidad en el lado del servidor. Quedaba pues la tarea de elegir un buen toolkit de generación de código. Se tuvieron en cuenta dos posibilidades ambas de código abierto, Apache AXIS y gSOAP. Apache AXIS nunca llegó a funcionar en 64 bits en la máquina de desarrollo; el proyecto AXIS está mal mantenido y las versiones actuales disponibles no son muy estables. Sin embargo, gSOAP es una herramienta que está muy bien considerada, eficiente y fácil de utilizar por lo que nos decantamos por ella.

gSOAP divide la generación de código a partir de un archivo WSDL en dos pasos:
wsdl2h [fichero WSDL]: Este comando genera un fichero de cabecera en formato C++ que puede ser más fácilmente editado en un IDE que un archivo WSDL y además al estar comentado en formato doxygen se puede generar documentación en html a partir de él.

soapcpp2 [fichero .h]: Este comando recibe como entrada el archivo de cabecera generado mediante wsdl2h y produce todo el código necesario para el cliente o servidor o ambos. Genera el .wsdl si este no existía.

Archivos más importantes que genera gSOAP son:

<nombre del servicio>.h

La cabecera que describe el servicio mediante una sintaxis propia de gSOAP. No se utiliza en la compilación; sólo en la generación de código.

<nombre del servicio>.wsdl

El wsdl que describe el servicio (generado por soapcpp2 al procesar un .h si el wsdl no existía

soapC.cpp

Contiene los serializadores y deserializadores de las estructuras de datos que viajan por el web-service

<nombre del servicio>.nsmap

Tabla de mapeado del servicio del namespace
http://www.fdi.ucm.es:8080/biblioteca.wsdl

soapH.h

Fichero que deben incluir tanto el cliente como el servidor. Por ejemplo contiene llamadas a los handlers del servidor, por lo que estos deben estar implementados en algún lugar

soapServer.cpp

Rutinas esqueleto de las llamadas atendidas por el servidor

soapClient.cpp

Rutinas esqueleto para realizar las llamadas al servidor

Soap<nombre del servicio>Proxy.h

Contiene una clase que permite hacer las llamadas al servicio desde el cliente a través de un objeto. Cada mensaje de petición del servicio puede ser enviado llamando a un método de este objeto.

Soap<nombre del servicio>Object.h

El equivalente al proxy pero para el servidor

5.3. Características del servicio desarrollado, el WSDL

El modelo de uso del servicio es document/literal. Al no ser RPC se define un tipo de XML Schema diferente para cada mensaje de forma que se pueda diferenciar cada operación. Por ejemplo una petición para SOM (Self Organizing Map) tiene definido el siguiente schema:

```
<!-- operation request element -->
 <element name="somRequest">
  <complexType>
   <sequence>
     <element name="dat" type="xs:base64Binary"/>
     <element name="wdist" type="xs:base64Binary" minOccurs="0" maxOccurs="1"/>
     <element name="cvin" type="xs:base64Binary" minOccurs="0" maxOccurs="1"/>
     <element name="parameters" type="engene:parameter" minOccurs="0" maxOccurs="15"/
   </sequence>
  </complexType>
 </element>
```

En el que se especifica que el mensaje llevará cuatro posibles elementos de datos, además de los parámetros que recibirá el programa de línea de comandos som. El mensaje de petición tendrá por tanto un parámetro de tipo somRequest.

```
<message name="somRequest">
 <part name="params" element="engene:somRequest"/>
</message>
```

A la hora de llamar al programa del paquete Eng*e*ne por línea de comandos se pueden especificar numerosos parámetros. Como la mayor parte son opcionales se prefirió no incluirlos en las peticiones en las peticiones como un parámetro obligatorio, ya que el desarrollador que escribiese un cliente del servicio debería especificar muchos parámetros hasta poder realizar la llamada. Además supondría una interfaz menos flexible. En su lugar, cada parámetro que se debe enviar al programa por línea de comandos se especifica en el elemento parameters, que es un array de tipo engene:parameter:

```
 <complexType name="parameter">
  <sequence>
   <element name="name" type="xsd:string"/>
   <element name="value" type="xsd:string"/>
  </sequence>
 </complexType>
```

Por ejemplo som recibe doce parámetros opcionales, tales como el tipo de distancia a utilizar o la topología, pero el cliente solo debe especificar los que quiera añadiéndoselos al array parameters en el mensaje somRequest.

5.4. El servidor

5.4.1. Arquitectura

El servidor está compuesto por un bucle infinito que crea un hilo phtread por cada nueva petición que recibe, de forma que puedan mantenerse en estado de proceso varias peticiones al mismo tiempo aunque haya menos procesadores disponibles que el número de peticiones. Los pthreads trabajan en detached mode, de forma que liberan sus recursos una vez acabados.

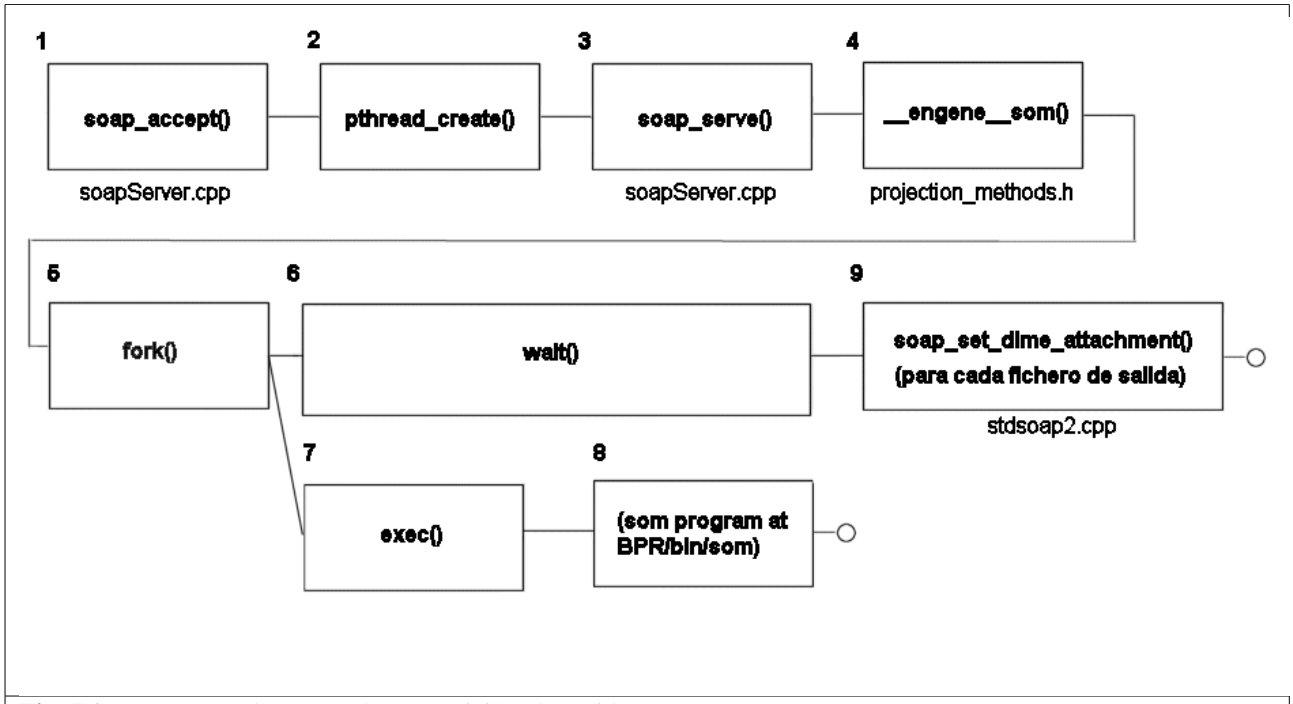

Fig. 5.1 Etapas para el proceso de una petición al servidor

Las etapas para despachar una petición som se pueden apreciar en el diagrama:

1. El servidor, en su bucle infinito, hace una llamada a soap_accept, un wrapper que proporciona gSOAP sobre la llamada UNIX accept. Al igual que la llamada UNIX, soap_accept puede ser bloqueante como es el caso: la función no retorna hasta que una conexión nueva se hacer al socket.
2. Se crea un nuevo hilo que procesará la petición.
3. soap_serve() elige el handler adecuado para el manejo de la petición y lo llama.
4. Se ejecuta el handler específico para procesar la operación som llamado, el handler recibe como parámetros string los argumentos que deben pasarse a som, tales como el tipo de topología o si los deben normalizarse o no.
5. Un nuevo proceso ha de crearse para ejecutar el programa som del paquete Engene.

6. El proceso pasa a convertirse en som.
7. El proceso padre espera a que acabe el hijo, de forma que cuando acabe pueda completar la petición.
8. Cada fichero que ha generado som se envía como attachment DIME.

5.4.2. Envío de ficheros

Los archivos son enviados en streaming, es decir que son consumidos (en este caso volcados a disco) mientras están siendo enviados. Los datos procedentes de los archivos llegan en pequeñas partes que son procesadas en funciones de C++ a las que gSOAP llama ante la llegada/envío de datos. Es algo similar a un evento; la llegada de una porción de datos o la posibilidad de envío dispara la ejecución de una función handler que lo maneja. En las funciones de llegada los datos son volcados a disco y en las de envío son lecturas a disco lo que se produce.

Cada pthread recibe unos ficheros como entrada y otros como salida. Como puede haber muchos hilos ejecutándose concurrentemente, no debe haber colisiones entre los nombres de los ficheros de diferentes hilos. Para ello, los ficheros que maneja cada pthread, reciben un nombre diferente en función del identificador del pthread, una estructura de tipo pthread_t; así los datos de entrada tienen el nombre data(*phtread_t*).*extensión* y los de salida output(*pthread_t*).*extensión*.

5.4.3. El log

El servidor mantiene además un fichero de log, error_log, donde se indican los sucesos que ocurren a lo largo del tiempo de forma que puedan identificarse más fácilmente las fuentes de problemas, como fallos en la especificación del servicio o en los handlers del servidor.

También posee un archivo de configuración donde se permite especificar el puerto donde escuchará las peticiones del servidor y además el directorio donde se encuentren los binarios del paquete engene.

5.4.4. Memory profiler

Todos los handlers de peticiones del servidor han sido analizados con un memory profiler, valgrind, para evitar memory leaks o comportamientos indefinidos como los causados por el uso de variables sin inicializar. Esto es necesario ya que el servidor podría estar ejecutándose sin ser reiniciado durante mucho tiempo y los memory leaks podrían acumularse hasta dejar sin memoria al servidor.

5.5 El cliente

5.5.1. Introducción

El cliente debía demostrar de forma minimalista las posibilidades que abría el web service de Engene. No es más que un frontend de los programas del paquete engene; apenas contiene lógica

de aplicación y es fundamentalmente interfaz gráfica.

Aunque se planteó la posibilidad de utilizar Java, finalmente se decidió utilizar C++ para aprovechar el código que gSOAP podía generar en el lado del cliente. El cliente crea un hilo cada vez que se indica la ejecución de un algoritmo del paquete engene, de forma que la interfaz no se queda congelada mientras se esperan los datos respuesta de ese algoritmo.

La interfaz gráfica está realizada en GTK. GTK es un "widget toolkit" para C (originalmente) aunque sus diseñadores utilizan un paradigma orientado a objetos [18] [19]. Para concentrarse más en la lógica de la aplicación y no tanto en el desarrollo de la interfaz gráfica, se utilizó glade (un editor visual) para la generación del código de la interfaz.

5.5.2. Principales archivos del cliente

La interfaz está separada de la lógica, los archivos relacionados con la interfaz gráfica están en interface.c, callbacks.c y visualization.cpp, mientras que en controller.cpp se concentra la lógica de la aplicación (que únicamente consiste en pasar los datos seleccionados por el usuario por una llamada servicio).

interfaces.c (generado por glade): contiene el código que crea las ventanas y asigna los eventos a unos manejadores.

calllbacks.c: contiene los callbacks que responden a eventos tales como el click del ratón.

controller.c: contiene las llamadas al proxy generado por gSOAP.

visualization.cpp: contiene funciones para mostrar las imágenes generadas por el paquete Eng*e*ne.

El proceso de realización de una petición som con nuestro cliente de prueba es:

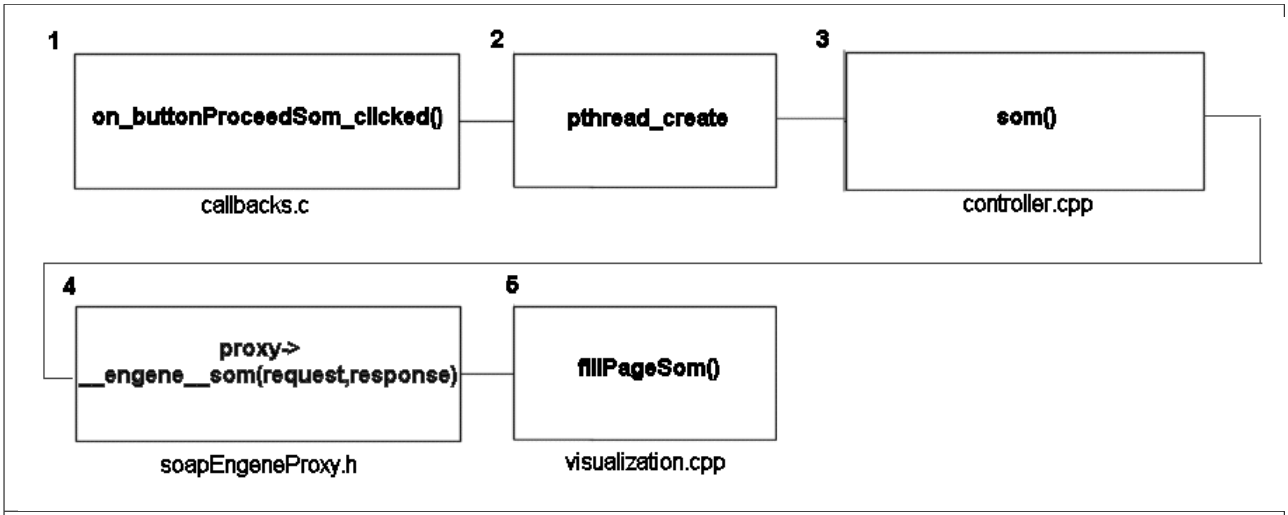

Fig. 5.2. Etapas para el proceso de una petición al servidor

1 y 2: El callback de la interfaz gráfica responde a un evento de ratón, recogiendo los datos la interfaz gráfica y pasándoselos a un nuevo hilo para que realice la petición. Si no se crease un nuevo hilo, el cliente se quedaría "colgado" hasta que la petición del servidor llegara (y la petición podría demorarse varias horas o incluso días).

3: La función som de controller.cpp pasa los parámetros que ha recibido de la interfaz.

4: Se ejecuta la función som del proxy generado por gSOAP.

5: Se rellena la interfaz con las imágenes (y el archivo .inf) generado por el paquete Eng*e*ne y que han llegado por el servicio.

5.6. Manual de uso

5.6.1. Servidor

5.6.1.1. Compilación

Pasos a seguir:

1 .- Descomprimir engene-server-src.tar.gz, que está en el raíz del CD

2 .- Pasarse al directorio engene-server y ejecutar

./configure

make

5.6.1.2. Instalación

Un vez compilado el servidor o descomprimido el tar de binarios, engene-server-x86-64.tar.gz que está en el raíz del CD, ejecutar:

make install

(Esto instalará el servidor en /usr/local/engene-server, para instalarlo en algún lugar diferente, cambiar la llamada a configure por configure –prefix=<directorio>).

5.6.1.3 Configuración

El servidor se configura con el archivo server.conf donde puede configurarse donde están los binarios del BPR de Eng*e*ne y el puerto donde debe escuchar el servidor. En caso de no existir tal fichero se utilizaran unos valores por defecto.

5.6.2 Cliente

5.6.2.1. Compilación

Pasos a seguir:

1 .- Descomprimir engene-client-src.tar.gz, que está en el raíz del CD.

2 .- Pasarse al directorio engene-client y ejecutar:

./configure

make

5.6.2.2. Instalación

Un vez compilado el servidor o descomprimido el tar de binarios, engene-client-x86-64.tar.gz ó engene-client-x86.tar.gz que están en el raíz del CD, ejecutar:

make install

(Esto instalará el servidor en /usr/local/engene-client, para instalarlo en algún lugar diferente, cambiar la llamada a configure por configure –prefix=<directorio>).

5.6.2.3 Configuración

El cliente se configura con el archivo client.conf donde se indica la dirección del servidor engene, ejemplo: http://marbore.dacya.ucm.es:8080 . En caso de no existir tal fichero se utilizaran unos valores por defecto.

6. Un caso de estudio real

Se parte del cliente del cliente del servicio web (Figura 6.1) en la cual se debe seleccionar una de las herramientas disponibles para su utilización.

Fig. 6.1 Ventana de Inicio del cliente del servicio web

Una vez seleccionada una de las herramientas, en este caso Bacth Som (Figura 6.2 y 6.3), se debe introducir un fichero de datos para realizar su posterior visualización.

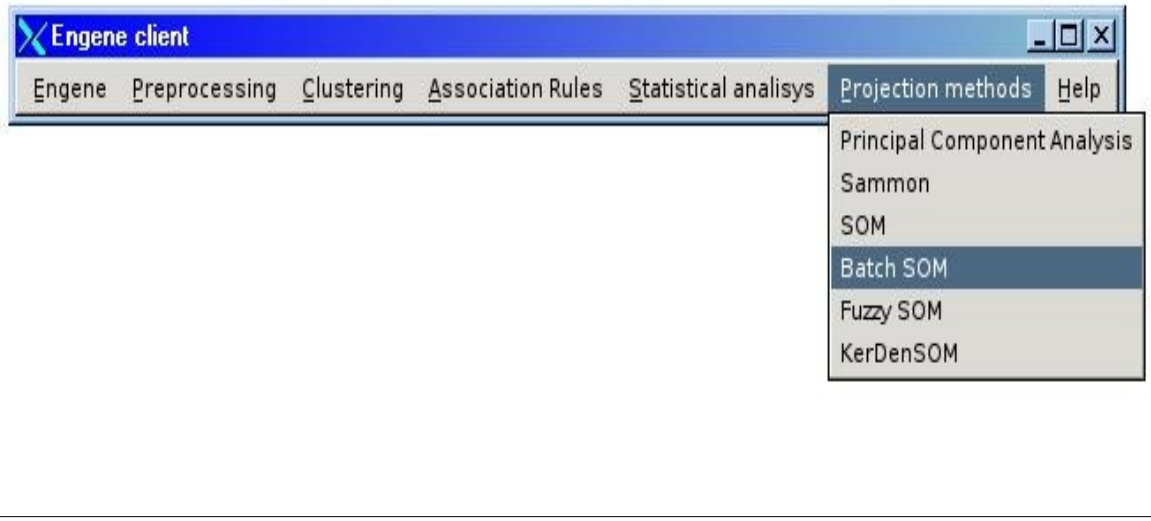

Fig. 6.2. Elección de Batch Som

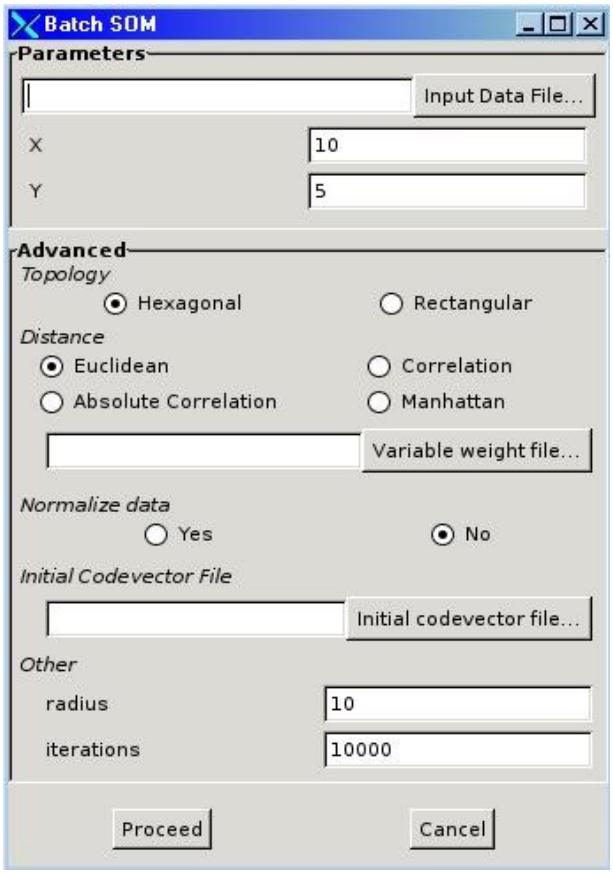

Fig. 6.2. Ventana de Batch Som

Para realizar la elección se pulsa sobre el botón llamada Input Data File, que nos muestra una ventana de búsqueda de archivos.

Una vez seleccionado el archivo, se deja el resto de valores por defecto y se pulsa sobre el botón Proceed.

Cuando Batch Som ha terminado de procesar el archivo, genera unas imágenes que son mostradas en la ventana de visualización (Figura 6.3).

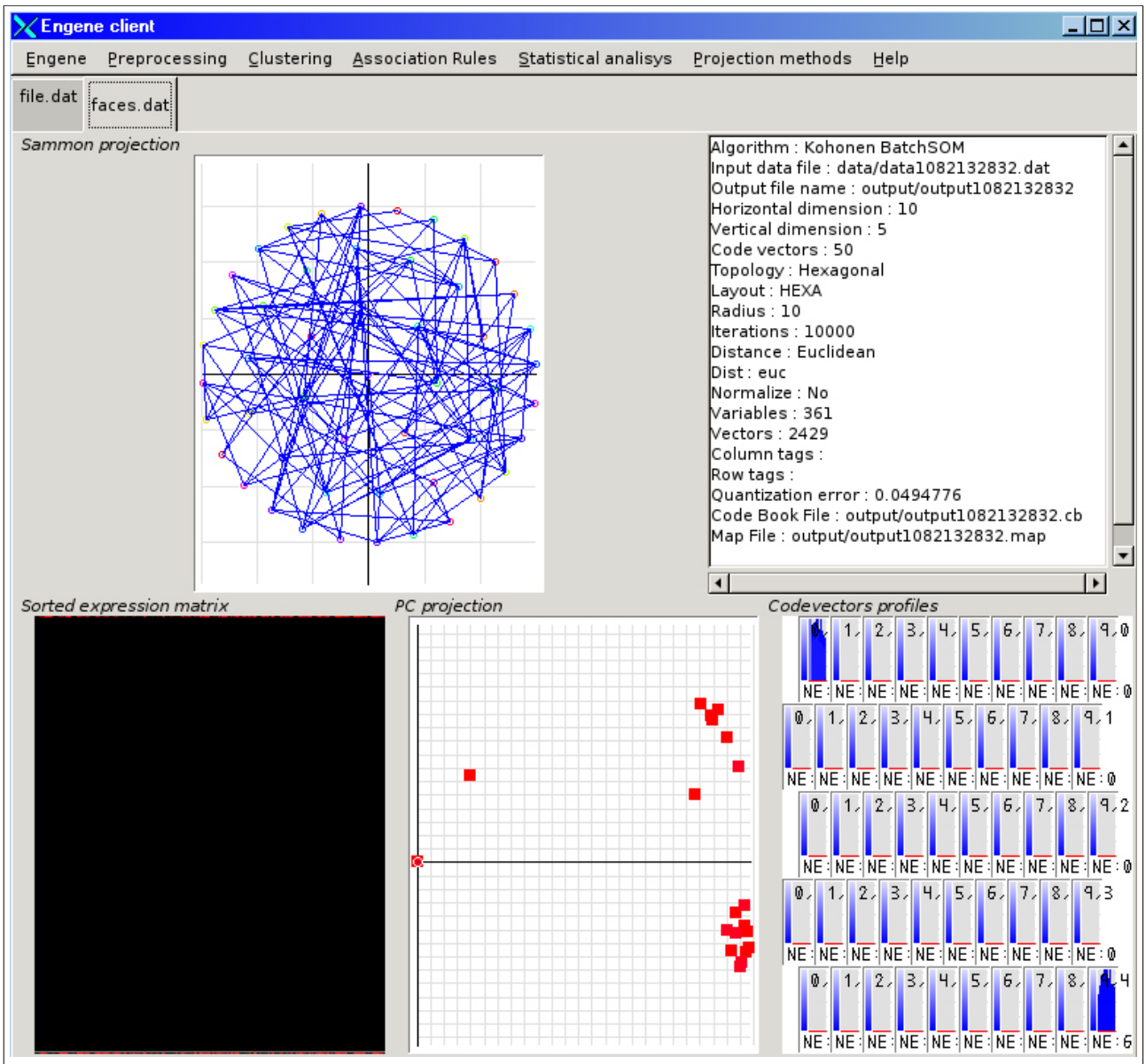

Fig. 6.3. Ventana con la visualización de imágenes a partir de datos por defecto

Por último se puede ver que ocurre si se cambia los valores por defecto de X a 20 y de Y a 10 se obtiene la figura 6.4

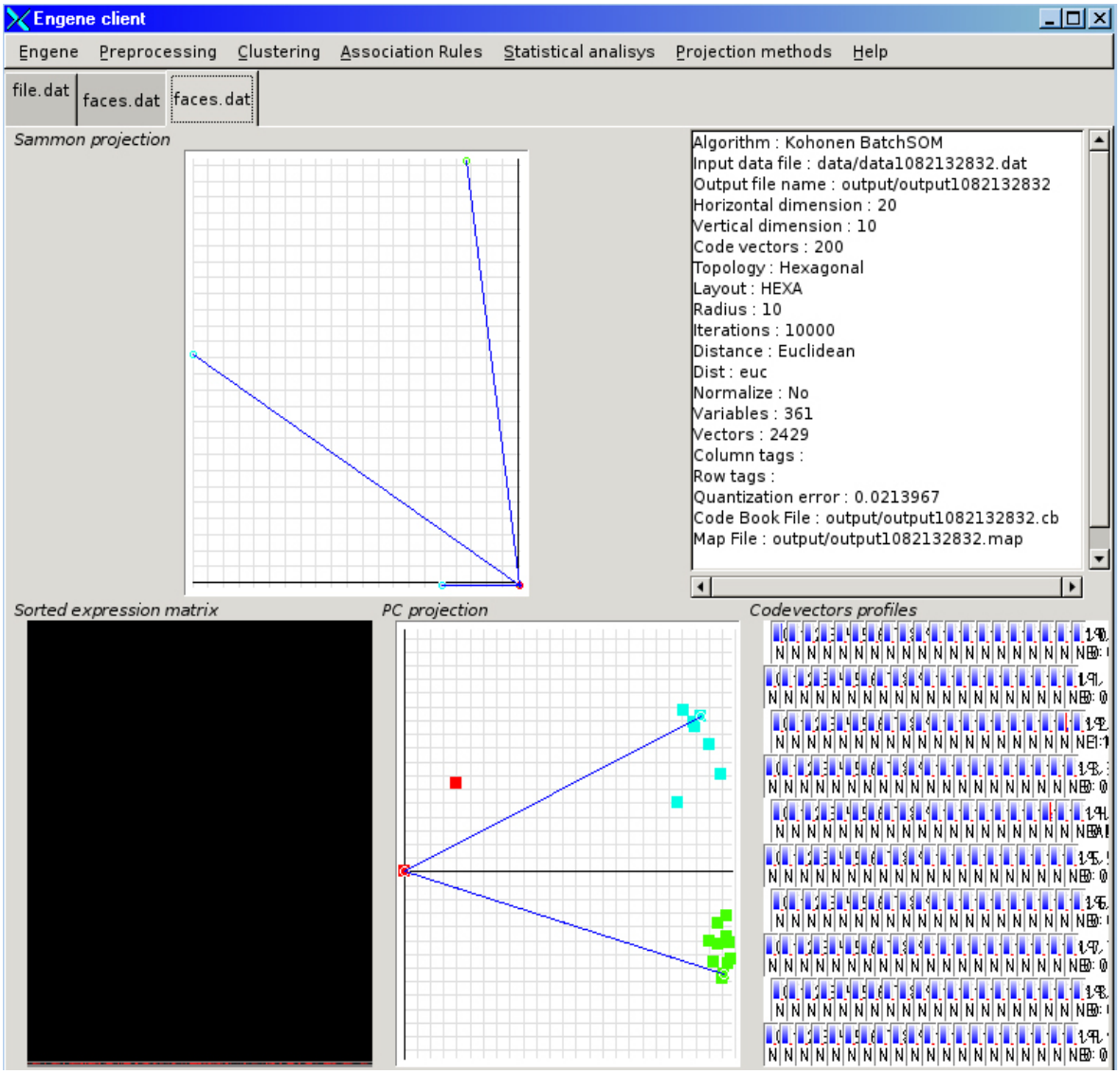

Fig. 6.3. Ventana con la visualización de imágenes a partir de datos por defecto cambiados para X a 20 e Y a 10

Se puede apreciar que todas las imágenes generadas han cambiado. Se puede resaltar que el número de neuronas representadas en codevector profieles es ahora de 200, frente a las 50 utilizadas a raíz de los valores por defecto, y que los genes representados por las proyecciones Sammon y PC aparecen más agrupados.

7. Conclusiones y trabajo futuro

Con este proyecto iniciamos una nueva etapa en el Eng*e*ne ya que creamos un servicio web en el que es posible ejecutar todos los algoritmos que existen en él de una forma distribuida. Esto lleva a no ser dependiente de una aplicación cliente o de un portal web, simplemente con conocer el funcionamiento del archivo wsdl se puede interactuar completamente con el servicio web.

Todo está diseñado para que de cara al futuro se pueda montar varios servidores distribuidos (Grid) a través de una red, por ejemplo de Internet, y con ello conseguir aumentar la capacidad de procesamiento.

MTOM es otro método de envío de ficheros adjuntos, reconocido como recomendación en el W3C y que pretende sustituir tanto a MIME como a DIME. Lamentablemente la versión disponible de gSOAP con la que comenzamos a escribir el servicio no la soportaba. El uso de MTOM por el servicio en un futuro podría mejorar la facilidad de creación de clientes para el servicio, ya que probablemente se convertirá en la corriente principal de envío de ficheros en los web services.

8. Bibliografía

[1] Bioinformatics : a practical guide to the analysis of genes and proteins / Andreas D. Baxevanis, B.F. Francis Ouellette.

Hoboken (New Jersey) : Wiley-Interscience, cop. 2005

[2] Introduction to bioinformatics / Arthur M. LeskLesk, Arthur M.

Oxford University Press, 2005

[3] Bioinformatics for dummies / Jean-Michel Claverie and Cedric Notredame.

New York : Wiley, cop. 2003

[4] Microarrays y biochips de ADN: Informe de Vigilancia Tecnológica / Marta López, Paloma Mallorquín, Miguel Vega

Genoma España, Octubre 2002

[5] Introducción a la bioinformática / T.K. Attwood, D.J. Parry-Smith

[6] AMIA Toolbox for MATLAB®. Copyright (c) 2004 Battelle Memorial Institute

[7] T. Kohonen, Seft-Organizing Maps, Second Edition, Springer-Verlag (1997).

[8] A Novel Neural Network Technique for Analysis and Classification of EM Single-Particle Images. A. Pascual-Montano, L. E. Donate, M. Valle, M. Bárcena, R. D. Pascual-Marqui, J.M. Carazo. Journal of Structural Biology, Vol. 133, No. 2/3, Feb 2001, pp. 233-245.

[9] Página de Engene ubicada en la Universidad Complutense de Madrid,

http://marbore.dacya.ucm.es:8081

[10] Página de Engene ubicada en la Universidad Autónoma de Madrid,

http://www.engene.cnb.uam.es

[11] Page of X-Window-based Microscopy Image Processing Package,

http://xmipp.cnb.csic.es

[12] Pattern-Recognition and classification of Images of Biological Macromolecules using Artificial Neural Networks. Marabini R., Carazo, J.M. Biophys J. 66:(6) 1804-1814 Jun 1994.

[13] Smoothly Distributed Fuzzy c-Means: a New Self-Organizing Map. Pascual-Marqui RD. Pascual-Montano AD. Kochi K. Carazo JM. Pattern Recognition 34:2395-2402, 2001

[14] Web services Essentials / Ethan Cerami 2002 O'Reilly

[15] Wikipedia – Web Service http://en.wikipedia.org/wiki/Web_service

[16] Wikipedia – Web Service Protocol Stack
http://en.wikipedia.org/wiki/Web_Services_Protocol_Stack

[17] gSOAP user guide http://www.cs.fsu.edu/~engelen/soapdoc2.html

[18] Wikipedia - GTK+ http://en.wikipedia.org/wiki/GTK+

[19] Developing Linux applications with GTK+ and GDK / Eric Harlow 1999 New Riders

[20] DIME Sending Files, Attachments, and SOAP Messages Via Direct Internet Message Encapsulation. http://msdn.microsoft.com/msdnmag/issues/02/12/DIME/

[21] BIND SOAP manual

http://soap.bind.ca/BINDSOAPManual.pdf#search=%22BIND%20Biomolecular%20Interaction%20Network%20Database%20manual%20DIME%