Ingeniería y Competitividad
ISSN: 0123-3033
inycompe@gmail.com
Universidad del Valle
Colombia

Barraza, Fernando; Salazar, Gustavo; Cuesta-Astroz, Yesid; Restrepo, Oscar E.
Implementación de una arquitectura web para la ejecución de flujos de trabajo en bioinformática
Ingeniería y Competitividad, vol. 8, núm. 2, 2006, pp. 34-45
Universidad del Valle
Cali, Colombia

Implementación de una arquitectura *web* para la ejecución de flujos de trabajo en bioinformática

Fernando Barraza* ° §, Gustavo Salazar* °, Yesid Cuesta-Astroz* &, Oscar E. Restrepo*

**Laboratorio de Bioinformática, ParqueSoft, Cali, Colombia*
°Escuela de Ingeniería de Sistemas y Computación, Universidad del Valle, Cali, Colombia
&Laboratorio de Biología Molecular y Patogénesis, Facultad de Salud, Universidad del Valle, Cali, Colombia
§ e mail: fbarraza@parquesoft.com

(Recibido: Mayo 2 de 2006 - Aceptado: Noviembre 10 de 2006)

Resumen

Con el advenimiento de las tecnologías para la generación masiva de datos de secuencias, en la bioinformática ha surgido recientemente la necesidad de automatizar la ejecución de las herramientas utilizadas para analizar e integrar la gran cantidad de información que es objeto de estudio por parte de sus usuarios. En este campo, una de las aproximaciones es la utilización de sistemas de flujos de trabajo. Al respecto, este trabajo analiza las características particulares de dichos flujos de trabajo, su función y su interrelación con los demás componentes de una plataforma de análisis bioinformático. Como resultado del análisis, se presenta una propuesta de arquitectura de software de una herramienta *Web* que permite la extensión funcional de diferentes sistemas de flujos de trabajo. Al final, se presentan los resultados de la implementación de un prototipo de software de la herramienta y la ejecución de una prueba de concepto en un caso real.

Palabras claves: Bioinformática, Ingeniería de software, Arquitecturas de software, Flujos de trabajo.

Implementation of a Web architecture for the execution of workflows in bioinformatics

Abstract

With the incoming of the massive data generation technologies in bioinformatics, the need to automate the execution of routine processes to analyze and integrate great amount of data has recently emerged. In this field, one approach is based on workflows. In this paper, specific workflow characteristics, and the functions and interrelations with all the other components of a bioinformatics platform, are discussed. As an outcome, a proposal of software architecture for a Web tool is presented, which allows the execution and functional extension of several workflow systems. Finally, we show both the implementation results of a software prototype for the tool and the execution of a proof-of-concept in a real case.

Keywords: Bioinformatics, Software engineering, Software architectures, Workflows.

1. Introduccion

La bioinformática es una disciplina que emergió a principios de la década pasada y en la cual los desarrollos de software se han enfocado en construir herramientas aplicativas, cada una para un tipo de análisis biológico en particular. La mayoría de dichas herramientas ha sido desarrollada desde una perspectiva meramente computacional, dada la necesidad principal de la implementación de algoritmos para resolver problemas específicos normalmente ligados a métodos de análisis, donde la ingeniería de software ha tenido poca relevancia (Ouzounis & Valencia, 2003).

De otro lado, el trabajo común de un bioinformático requiere de la utilización de múltiples herramientas vistas como un conjunto lógico relacionado con el proceso analítico a realizar, lo cual impone la necesidad de la interoperabilidad entre las herramientas de forma que brinden una solución más completa. Dicha interoperabilidad, cuyo objetivo principal es la obtención e integración de datos biológicos con los sistemas y bases de datos existentes en los laboratorios, típicamente no se encuentra automatizada con lo cual la calidad de los resultados recae en el *know-how* y pericia en el uso de las herramientas informáticas por parte del bioinformático y no parte de un ejercicio de buenas prácticas o procedimientos estándares.

Estos hechos han vuelto la mirada a como se debe afrontar el desarrollo de software de las herramientas bioinformáticas para lograr su integración. Las organizaciones han entendido ésto y han empezado a implementar plataformas de análisis teniendo como eje principal los sistemas de flujos de trabajo, pero en muchos casos, sus aproximaciones están basadas en conceptos de arquitecturas de software tradicionalmente usadas en ambientes de aplicaciones empresariales. Estos modelos de arquitectura deben ser revisados y ajustados de forma que ofrezcan la flexibilidad, expresividad y el poder de transformación requerido para manejar las tareas de los bioinformáticos en un ambiente donde la plataforma sea robusta y heterogénea.

En el presente trabajo, presentamos una propuesta de arquitectura para una plataforma de análisis bioinformática implementada por el *Laboratorio de Bioinformática de ParqueSoft* (labs.parquesoft.com) para el *Centro Internacional de Agricultura Tropical* (CIAT, www.ciat.cgiar.org).

1.1 Antecedentes

En la unidad de biotecnología del CIAT se ejecutan tareas rutinarias de análisis de secuencias de ADN (Acido desoxirribonucleico) y proteínas de diversas especies vegetales tales como *oryza sativa* (arroz) y *manihot esculenta* (yuca) entre otras. Una de estas tareas es el análisis de *ESTs* (*Expressed Sequence Tags*). Los *ESTs* son pequeños segmentos secuenciados a partir de clones de ADNc (ADN complementario). Un *EST* se obtiene mediante una sola secuenciación automática y parcial de uno de los cientos de clones seleccionados al azar de una genoteca (base de datos de genes) de ADNc. Los *ESTs* sirven para descubrir genes desconocidos, mapear un genoma o identificar las regiones codificantes de éste.

Para estos análisis se había diseñado un flujo secuencial de análisis (Soto et al., 2004).implementado posteriormente en un programa en lenguaje *Perl*. Dicho programa ejecuta de forma sucesiva y controlada cada una de las herramientas bioinformáticas que implementan cada paso del análisis. En cada paso del programa, se ejecutan en forma explicita o código duro las herramientas bioinformáticas y, para unir todas las herramientas en un solo análisis, la entrada de cada herramienta es la salida de su predecesora, en donde, además, para cada ejecución, los parámetros requeridos están también explícitamente definidos.

Si bien esta forma de implementación cumple con las necesidades básicas de algunos análisis, un flujo de trabajo implementado de esta manera restringe la posibilidad de extender la funcionalidad del análisis durante su configuración mediante la inclusión de nuevas herramientas bioinformáticas o la eliminación de otras en cada instancia y, ya en tiempo de ejecución, la modificación de los parámetros

requeridos con sus valores y la toma de decisiones sobre flujos alternos, dependiendo de los resultados obtenidos durante el análisis. Igualmente, se observaron otros problemas con esta implementación como son la poca amigabilidad de la línea de comandos para la ejecución (lo cual originaba confusión en la entrada de los valores de los parámetros), la dificultad para manipular los archivos de resultados y la no integración con el LIMS (*Laboratory Information Management System*) existente en el laboratorio.

Todas estas características de la situación encontrada correspondían a la situación típica de un laboratorio analítico en el área de las biociencias que empieza a utilizar la bioinformática como un elemento principal en sus procesos de investigación. Con estos antecedentes, el CIAT, consciente de esta necesidad, determinó avanzar en la implementación de una plataforma de análisis para bioinformática cuyo objetivo principal fuera fortalecer los procesos de "analizar el gran número de secuencias para entender y descubrir la información en términos de estructuras de proteínas, funciones y evolución" (http://www.ciat.cgiar.org/biotechnology/bioinformatics.htm#purpose).

Los proyectos para el estudio de genomas parten de una fase de secuenciamiento en la cual se generan en el laboratorio lo que podríamos llamar datos brutos, los cuales son datos sin analizar. Sin llevar a una herramienta o programa bioinformático, esto se lleva a cabo con el fin de darle sentido biológico a este tipo de datos. Son estos datos secuencias de ADN, RNA o de proteínas entre otros. Este tipo de moléculas posee un papel fundamental en todos los organismos y se espera que al lograr entender bien este tipo de código se dé una revolución en varias áreas como la medicina, biología y agricultura, entre otras.

El gran desafío de los investigadores consiste en analizar estas secuencias y obtener información biológicamente relevante. Durante estos análisis, los investigadores utilizan diversas herramientas bioinformáticas, programas y un gran volumen de información almacenada en fuentes de datos de biología molecular. De hecho, uno de los objetivos de la bioinformática es que ante el volumen creciente de información se facilite el análisis por parte del investigador por medio de la creación de nuevos procesos en bioinformática.

Un flujo de trabajo lleva a la automatización del procedimiento de modo que documentos, bibliografía e información sobre tareas son distribuidas entre los participantes de la investigación. En conclusión, un flujo de trabajo administra el trabajo con secuencias, además de lograr automatizar una serie de programas que procesan datos experimentales y que ayudan al investigador a interpretar y analizar los datos.

En el campo de los flujos de trabajo en bioinformática se cuentan con distintos desarrollos como son:

Taverna: el proyecto *Taverna* apunta a proveer un lenguaje y una herramienta de software que faciliten el uso de flujos de trabajo y tecnología de computación distribuida de la comunidad *eScience*. Es un componente del proyecto *MyGrid* y está disponible gratuitamente bajo los términos de la licencia LGPL.

Taverna Workbench: permite a los usuarios construir complejos flujos de trabajo de análisis de componentes tanto remotos como locales sin conocer los detalles técnicos que cada tarea implica, ejecutando estos flujos de trabajo con sus propios datos y visualizando los resultados.

Pegasys: es un software flexible, modular y personalizable que facilita la ejecución e integración a partir de datos heterogéneos de herramientas utilizadas para el análisis de secuencias biológicas. En *Pegasys* se introduce una nueva estructura de datos para crear flujos de trabajo de análisis de secuencias y un modelo de datos unificados para almacenar los resultados.

Wildfire: es una interfaz gráfica para construir y correr flujos de trabajo. Usa las características de la interfaz de *jemboss* y adicionalmente una interfaz *drag and drop* permitiendo al usuario utilizar la suite de herramientas de EMBOSS dentro de los flujos de trabajo. GEL (*Grid*

Execution Language). *Wildfire* permite al usuario visualizar la construcción del flujo de trabajo para su ejecución. *Wildfire* exporta el flujo de trabajo como un *script* de GEL y llaman un intérprete de GEL para ejecutar éste. El intérprete puede correr desde la misma máquina o en un servidor remoto.

2. Metodologia

Como hemos mencionado, se requería de la implementación de una plataforma de análisis bioinformático, por lo que se inició con la conceptualización de una arquitectura robusta, flexible, amigable al usuario y extensible. Dicha plataforma debía basarse en las herramientas bioinformáticas existentes, brindando un nivel de abstracción que facilitara la inclusión de herramientas futuras y en cuyos servicios principales contemplara la integración de los sistemas y bases de datos existentes, además de la utilización de fuentes de datos públicas conocidas disponibles en *Internet*. Otro requerimiento importante era que la plataforma fuera flexible en la definición, reutilización y ejecución de los flujos de trabajo. Con todos estos requerimientos, era necesario determinar las características más relevantes de una plataforma de análisis en bioinformática para así poder plantear una arquitectura de la solución, determinando los componentes de software a desarrollar o reutilizar para su implementación.

2.1 Consideraciones sobre una plataforma de análisis en bioinformática

Una plataforma de análisis se puede concebir como un conjunto de tres componentes funcionales integrados: fuentes de datos, sistema de flujo de trabajo y herramientas de visualización (www.alphaworks.ibm.com/tech/biowbi). A continuación, describimos y analizamos cada componente.

2.2 Fuentes de datos

Es el componente que permite constituir fuentes de datos a partir de una o más bases de datos públicas especializadas (e.g. NCBI, www.ncbi.nlm.nih.gov) o las privadas catalogadas localmente. Una fuente de datos es un conjunto de datos que el bioinformático puede usar para sus análisis. Típicamente, este componente se encuentra descentralizado en un laboratorio de bioinformática, donde cada usuario mantiene sus propios archivos de datos catalogados de manera particular. Igualmente, las referencias a bases de datos públicas frecuentemente son contextualizadas de acuerdo al análisis en cuestión, reduciendo las posibilidades de ser reutilizadas. Otros laboratorios con mejor infraestructura tecnológica cuentan con software LIMS como medio de almacenamiento de los objetos de experimentación, que en el caso de la bioinformática son las secuencias de ADN, proteínas u otras estructuras moleculares. El LIMS también interviene en el proceso de anotación, documentación y clasificación tanto de los objetos de experimentación como de los resultados sobre los análisis hechos a los mismos.

2.3 Sistema o motor para la construcción de flujos de trabajo

Es el componente que permite coordinar la ejecución de manera controlada de cada una de las herramientas involucradas en el análisis siguiendo un orden lógico pero sensible a los diferentes modos de ejecución de dichas herramientas. Cada herramienta implementa uno o más algoritmos bioinformáticos, los cuales a su vez tienen múltiples parámetros que normalmente se especifican en una línea de comandos para su ejecución.

Los flujos de trabajo en bioinformática pueden resultar complejos debido a que los datos necesitan muchas transformaciones de un algoritmo a otro. Dichas transformaciones pueden tipificarse como sintácticas y semánticas (García-Castro et al., 2005). En las transformaciones sintácticas, el objetivo es meramente operacional y lo que se busca es poder pasar información de un algoritmo a otro de forma que exista una correspondencia lógica entre las herramientas que lo implementan. En las transformaciones semánticas, el aspecto sintáctico es separado del dominio del conocimiento utilizando una abstracción en la cual cada algoritmo tiene un significado independiente de la herramienta que lo implementa. Allí, es importante que el motor de flujos de trabajo soporte como mínimo un

conjunto de funciones de transición de estado entre cada una de las ejecuciones de las herramientas que implementan los algoritmos bioinformáticos. Estas funciones esenciales son, la *iteración* (ejecutar automáticamente el mismo algoritmo más veces con diferentes datos de entrada preservando los mismos valores en sus parámetros), la *recursividad* (ejecutar automáticamente el mismo algoritmo más veces sobre los mismos datos de entrada, variando el valor de un parámetro específico de un conjunto de valores definidos por el usuario), la *secuencialidad* (ejecutar de manera secuencial dos o más algoritmos usando la salida del primero como entrada del siguiente y creando lo que se conoce como una tubería), el *condicional* (ejecutar un algoritmo dependiendo de la evaluación de una condición lógica basada en los datos de entrada o de salida) y la función de *parada* (posibilidad de detener el flujo de ejecución en un punto preciso para poder evaluar los resultados obtenidos y decidir si continuar o no).

Además de las funciones de transición de estado, el motor debe comportarse de acuerdo a las reglas heurísticas definidas a través de un lenguaje que especifique dichas funciones y su interacción. La escogencia del lenguaje que soporte el flujo de trabajo está ligada necesariamente a las opciones de aquellos que sigan estándares abiertos ajustables a las necesidades en la bioinformática. El marco teórico está definido por los lenguajes de modelado de procesos (BPM), donde han ido apareciendo propuestas orientadas a servicios en la *Web*. Se destacan algunos estándares propuestos en www.ebPML.org como *Xlang* y WSFL y más recientemente BPEL, patrocinado por *Microsoft* (www.microsoft.com) e IBM (www.ibm.com), XPDL por la WfMC (*Workflow Management Coalition*, www.wfmc.org), extensiones a UML propuestas por OMG (*Object Management Group*, www.omg.org) y WSCI, el cual está bajo la supervisión de W3C (*World Wide Web Consortium*). Sin embargo, estos lenguajes, considerados en la categoría de lenguajes de orquestación de servicios (WSO) en su mayoría, apenas recientemente están siendo considerados para la implementación de *flujos de trabajo* bioinformáticos en plataformas de análisis. *MyGrid* es un ejemplo de estas plataformas de análisis (Stevens et al., 2003). Una razón es que las herramientas de alta calidad libres (*open source*) disponibles en bioinformática no soportan los lenguajes estándares comerciales del mercado. A su vez, la oferta de herramientas informáticas bajo plataformas propietarias limita la posibilidad de uso de muchos usuarios ya que por su costo y requerimientos de hardware hace difícil su implementación como herramienta de escritorio. Pero tal vez existen dos limitantes en el uso de las herramientas del mercado en un entorno bioinformático: la flexibilidad para representar los flujos de trabajo y su robustez para ejecutarlos. En el primer caso, los lenguajes para la representación de servicios *Web* lo hacen a un nivel de implementación, lo cual limita la posibilidad de abstraer en tareas genéricas aquellas que implementan cada tipo de análisis frecuentemente requeridos para el trabajo en bioinformática (Addis et al., 2003). Igualmente, y aunque los servicios *Web* desde su aparición han venido madurando en su robustez, en esencia son de naturaleza sincrónica mientras que muchos de los flujos de trabajo en bioinformática requieren de asincronismo para poder ejecutar algoritmos de larga duración, al mismo tiempo que se efectúan otras tareas.

En resumen, los flujos de ejecución en bioinformática, a diferencia de otros campos, presentan de una alta variabilidad del tipo y los resultados de los análisis, lo cual los hace difícil de mantener y extender. Algunas propuestas de flujos de trabajo que hacen parte de iniciativas como *BioMoby* (www.biomoby.org) y *MyGrid* (www.myGrid.org) intentan resolver este problema desde la perspectiva de no estandarizar los datos sino su integración (Wilkinson & Links, 2002). Cabe anotar que *BioMoby* y *MyGrid* no son motores de flujo de trabajo: *Biomoby* es una arquitectura para servicios web bioinformáticos y *MyGrid* es un entorno de trabajo que soporta además otro tipo de actividades y que utiliza un sistema de flujos de trabajo llamado *Taverna* (taverna.sourceforge.net).

2.4 Herramientas de visualización

Proveen el acceso a la información relativa a los análisis que se han enviado a ejecución en el sistema de flujos de trabajo. Se puede evaluar el estado de la ejecución y revisar el resultado

concerniente a análisis previos. Para la implementación de estas herramientas, una arquitectura orientada a servicios (SOA) es una de las alternativas interesantes a considerar. Este modelo ofrece la posibilidad de que se conecten las herramientas a servicios de representación de información ya sean locales o remotos, además de su independencia del motor de flujo de trabajo. Revisando la literatura existente y la disponible en *Internet*, es notoria la ausencia de referencias a SOA por parte de las comunidades bioinformáticas que desarrollan herramientas de código abierto. No obstante, servicios *Web* no es lo mismo que SOA. Existen otros mecanismos como RMI y JMS, entre otros, para su implementación. En últimas, lo importante es que cada herramienta y el conjunto de ellas sigan un patrón de arquitectura claramente definido que permita su fácil integración a la plataforma.

De otro lado, estas herramientas poseen en su mayoría interfaces gráficas poco amigables, donde la ausencia de patrones de facilidad de uso dificulta la apropiación de la experiencia del usuario y que su trabajo con la plataforma de análisis sea cada vez más productivo. La tendencia en la extracción, representación y despliegue de información en este tipo de herramientas, es limitarla al contexto del análisis ejecutado, lo cual genera problemas como la ausencia de un significado semántico de los resultados obtenidos desde un sistema de flujo de trabajo. En términos prácticos, un bioinformático tiene que navegar *a* través de más información de la requerida dedicando más esfuerzo a encontrar resultados importantes en su investigación (Vailaya et al., 2004). Para abordar este problema, debemos considerar como opción la utilización de generadores automáticos de interfaces que ejecuten las herramientas bajo un patrón *wrapper* (envoltorio) , presentando una interfaz de usuario más adecuada con los requerimientos planteados. Las interfaces a su vez tendrían la posibilidad de extender los servicios de la plataforma y generarse dinámicamente a partir de ontologías predefinidas. La aproximación a esta implementación está ligada al campo de la *Web* semántica y de los lenguajes de definición de ontologías como OWL (*Ontology Web Lenguaje*), entre otros.

2.5 Diseño de una arquitectura adecuada para la plataforma

Dadas las condiciones y características particulares de una plataforma bioinformática y entendiendo la necesidad de que su implementación sea flexible y escalable, se estableció una arquitectura donde se siguieron principios básicos determinantes, tales como la independencia de sus componentes, la posibilidad de integrar nuevas fuentes de datos y herramientas de análisis, la selección de un motor para la construcción de flujos de trabajo con la capacidad suficiente para ejecutar los análisis requeridos y una capa de servicios que enmarquen las diferentes herramientas de visualización e intercambio de información de la plataforma.

El resultado fue una arquitectura que sigue un modelo por capas (almacenamiento de datos, flujo de trabajo, visualización) que permite ir desde la adquisición de los datos con los equipos del laboratorio (*wet lab*) hasta la presentación de los resultados de los análisis. La solución integra los sistemas existentes (ej. LIMS) y las bases de datos bioinformáticas tanto públicas como privadas.

De acuerdo con la arquitectura definida, se determinaron sus elementos. Se destacan en la plataforma la utilización de un motor de flujos de trabajo (en el cual se definen y ejecutan los flujos de trabajo) y el desarrollo de una herramienta de interfaz que, independientemente del motor de flujo de trabajo utilizado, lo extiende funcionalmente y a la vez integra los demás componentes concebidos en la solución. Por ejemplo, el motor de flujo de trabajo seleccionado es extendido funcionalmente al agregarle la capacidad a la plataforma de un control de sesiones de usuarios que permita asociar los parámetros, resultados obtenidos y estructura del flujo de trabajo que se ejecuta, al propietario de la información. En el caso de la integración de información, el sistema LIMS preexistente se integra de forma que los datos de entrada de los análisis en la plataforma provienen de él y sus resultados pueden ser almacenados de vuelta.

En el contexto de este artículo, hacemos referencia al motor de flujo de trabajo seleccionado y nos

concentramos en el diseño de la herramienta *Nirvana* como interfaz gráfica de la plataforma.

2.6 Selección de un motor de flujos de trabajo

Para la selección del motor de flujo de trabajo a utilizar en la plataforma se revisaron varios de los más conocidos y utilizados en el ambiente bioinformático. Se evaluaron sistemas tales como PISE (www.pasteur.fr/recherche/unites/sis/Pise), que provee algunas herramientas que permiten combinar métodos de análisis. El usuario puede definir un programa usando *Bioperl* (www.bioperl.org) o usar una interfaz donde puede registrar la macro resultante. En cualquier caso se asume que el usuario sabe programar en *Perl*. Otra opción evaluada fue *Biopipe* (www.biopipe.org) la cual provee integración para algunas herramientas analíticas usando una API (*Application Program Interface*) hacia *Bioperl* y almacenando los resultados en una base de datos MySQL lo cual permite al usuario hacer algún tipo de monitoreo sobre la ejecución del flujos de trabajo. El proyecto *Taverna* que sigue un enfoque hacia *MyGrid* basado en la utilización de servicios *Web*, provee capacidades superiores a las ofrecidas por los anteriores mencionados; sin embargo, no posee un generador de interfaces gráficas que permita fácilmente la inclusión a la plataforma de nuevas herramientas de análisis. *Pegasys* (bioinformatics.ubc.ca/pegasys), otra alternativa, es una iniciativa muy similar a *Taverna*.

El motor seleccionado fue GPIPE (http://if-web1.imb.uq.edu.au/Pise/5.a/welcome.html) el cual se ejecuta sobre PISE. GPIPE provee la capacidad a los usuarios de definir y compartir flujos de trabajo completos a varios niveles de información reduciendo la complejidad sintáctica que esto involucra (García-Castro, 2004). GPIPE provee una implementación que está construida sobre una estructura sintáctica flexible y un conjunto de operadores algebraicos para flujos de trabajo analíticos. GPIPE ejecuta las herramientas localmente, lo cual permite tener un control centralizado de ellas y no depender de servidores remotos y del comportamiento variable de la red para tener una correcta ejecución del análisis.

3. Resultados y discusión

Nirvana es una interfaz *Web* escrita en *Java*, utilizando recomendaciones y estándares J2EE, que permite la configuración y utilización de motores para la ejecución de flujos de trabajo bioinformáticos y la administración de las ejecuciones de los análisis realizados sobre éstos. La arquitectura con que ha sido concebido *Nirvana* ofrece además la posibilidad de suscribir diferentes servicios de visualización y representación de información. *Nirvana* tiene licencia GPL y su arquitectura está concebida de manera abierta como parte de la iniciativa de desarrollar un conjunto de herramientas que son objeto de los proyectos de investigación aplicada y desarrollo (IA+D) del *Laboratorio de Bioinformática de ParqueSoft*.

En esta, su primera versión, *Nirvana* utiliza GPIPE como motor de ejecución de flujos de trabajo. *Nirvana* fué diseñada con la intención de ofrecer una mejor interfaz de usuario que mejore la amabilidad (*user friendliness*) de la interacción respecto a las típicas interfaces de línea de comando extendiendo las capacidades actuales ofrecidas por GPIPE en su componente de visualización. Dichas funcionalidades adicionales son la capacidad para el manejo de sesiones de usuario, el almacenamiento de los resultados de las ejecuciones de los análisis y la posibilidad de realizar anotaciones sobre los mismos y sobre las diferencias en los valores utilizados en los parámetros de ejecución del flujo de trabajo. Lo anterior, le permite al usuario tener toda la información de las ejecuciones realizadas para un proyecto de manera organizada y así obtener mayores probabilidades de éxito en el análisis de los resultados, lo que se constituye en un repositorio centralizado de ejecuciones como puntos de referencia claves para futuros análisis. La herramienta cuenta además con una API (*Application Program Interface*) para mover datos de entrada y salida de los análisis a otras herramientas o sistemas (EJ. LIMS).

3.1 Arquitectura de *Nirvana*

Uno de los requerimientos de *Nirvana* es la de ser independiente del motor de flujo de trabajo de la

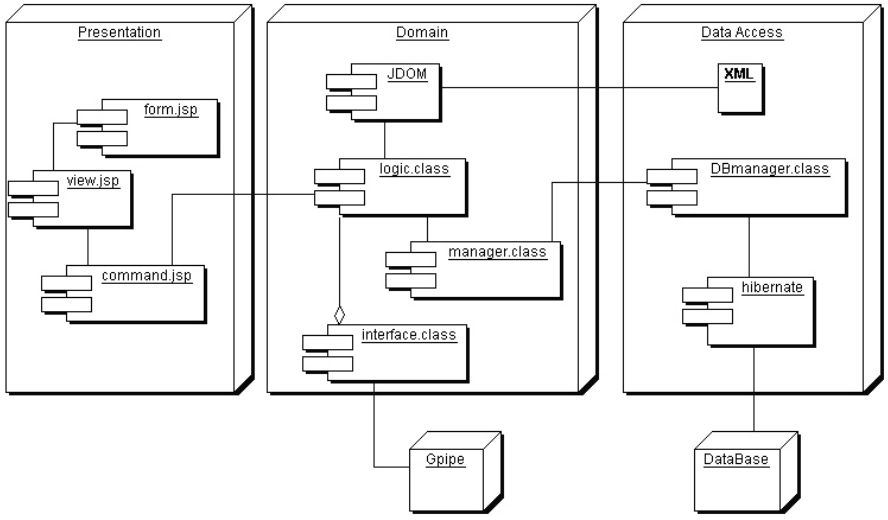

Figura 1. Arquitectura de software de Nirvana.

plataforma de análisis y de los servicios de almacenamiento y representación de información. Para lograr esa independencia, el enfoque de arquitectura abierta escogido, debía cumplir que la interfaz de usuario estuviera claramente diferenciada de la lógica de dominio y del acceso a los datos. En consecuencia, tomando como referencia el patrón de arquitectura MVC (modelo-vista-control) se definió la arquitectura de software para la herramienta según se muestra en la Figura 1.

En este esquema, se observa la típica arquitectura de tres capas que siguen un patrón MVC: la lógica de presentación (encargada de manejar la interacción entre el usuario y el software), la lógica de control (encargada de representar las labores que debe desarrollar la herramienta para la funcionalidad del dominio) y los mecanismos de acceso a la base de datos (encargados de la comunicación con los otros sistemas que desarrollan tareas para la herramienta como los son las bases de datos).

Figura 2. Interfaz gráfica de Nirvana.

En detalle, cada una de las tres capas cumple una función principal:

La capa de presentación brinda una independencia entre la forma en como son representados los resultados de los análisis y la lógica de ejecución de los flujos de trabajo. La interfaz del usuario le permite navegar lógicamente entre los flujos de trabajo, los resultados de sus ejecuciones y los datos anotados de los mismos (ver Figura 2). El patrón MVC facilita la inclusión de JSP como lenguaje *middleware* (intermediario) separando el código encargado del despliegue de las páginas (clases view.jsp y form.jsp) del responsable de la comunicación con la capa de dominio (clase command.jsp).

La segunda capa, que constituye el dominio de la aplicación, se implementó utilizando los conceptos del paradigma *orientado a objetos*, con el propósito de lograr que el núcleo de la aplicación sea absolutamente independiente de la capa de presentación, del acceso a los datos y del motor de flujo de trabajo.

Los motores usados para la construcción de flujos de trabajo bioinformáticos más reconocidos en la actualidad y revisados en el proyecto, utilizan documentos XML para la definición de sus flujos de ejecución, los cuales son el punto de conjunción del que se vale esta arquitectura para lograr comunicarse con dichos motores sin depender de un motor en particular. Para acceder, manipular y generar documentos XML, se utilizó el API JDOM (*Java Document Object Model*), el cual es instanciado desde varias de las clases lógicas de la herramienta (clase logic.class), que son las encargadas de las características funcionales propias. Para realizar la interfaz con el motor de flujos de trabajo se implementó una clase base, de la cual deben de heredar las clases que controlen cada motor. Para el caso implementado con GPIPE, fue necesario crear una librería en *Perl*, que aísla la ejecución de los flujos de trabajo de la interfase gráfica de este motor. Esta es la librería que *Nirvana*, en el caso particular de GPIPE, invoca mediante la clase de interfaz genérica. Adicionalmente se crearon unas clases (Manager.class), a modo de *Java beans* para la comunicación con la capa de acceso a los datos.

La tercer capa de acceso a datos, simplifica la forma de acceso y almacenamiento de éstos desde los servicios de análisis, independientemente de la base de datos utilizada. Para lograr este cometido, se utilizó *hibernate* (www.hibernate.org), un mapeador objeto-relacional para el lenguaje de programación *Java*, que permite tener acceso a diferentes bases de datos utilizando objetos de tipo *bean*, en lugar del tradicional SQL. Con esto, se logra que toda la herramienta soporte la programación orientada a objetos.

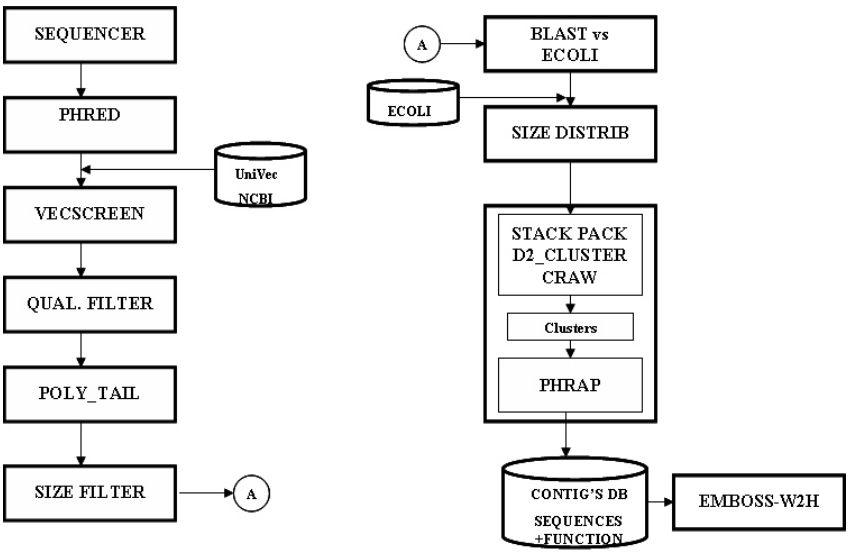

Figura 3. Flujo de trabajo para análisis de ESTs.

A su vez, los datos también pueden ser almacenados y cargados como documentos XML, para lo cual se hace uso de nuevo de JDOM. Para la interfaz entre el mapeador seleccionado y el resto de la herramienta la arquitectura implementa una clase independiente (clase DBManager.class). Las herramientas componentes de la plataforma fueron instaladas en un servidor *Linux* en el CIAT. Se configuró el flujo de trabajo en el motor GPIPE de acuerdo al flujo de trabajo escogido como piloto (Figura 3) que tiene como objetivo hacer análisis de *ESTs*. Posteriormente, se inició el proceso de pruebas con una población determinada de secuencias la cual debía ser una cantidad suficiente que pudiera ser considerada como evaluador de la calidad funcional y de rendimiento de la plataforma.

Es necesario considerar para el proceso de prueba, que para hacer un análisis de *ESTs* de gran magnitud, es crucial preparar los datos previamente de forma muy cuidadosa y exhaustiva de manera que se minimice el riesgo de incorporar datos (secuencias) que no estén relacionadas con el organismo con el que se esta trabajando. Es así como en estos procesos, las secuencias que se obtienen al final del proceso de amplificación, clonación y posterior secuenciamiento, pueden estar contaminadas con vectores de clonación, siendo éstos de origen bacteriano y convirtiéndose de esta forma en un ruido potencial para el análisis final de las secuencias. Esto debido a que existe la posibilidad de que alguna secuencia del vector pueda contaminar los resultados, a menos que se tomen medidas para identificarla y eliminarla, como es en el caso del flujo de trabajo implementado en nuestro piloto, donde los *ESTs* son sometidos a un proceso de control de calidad para eliminar contaminación por vector.

En este caso particular, se utiliza la herramienta llamada *Vecscreen* del NCBI la cual lleva a cabo una ejecución del algoritmo BLAST de la secuencia *Query* (secuencia de entrada a comparar) contra una base de datos de vectores. El siguiente paso en el flujo de trabajo es efectuar un control de calidad para secuencias poli A, poli T y poli C-T. No obstante ante la posibilidad de sufrir una contaminación por *E.coli* la cual es la bacteria donde se lleva a cabo la clonación del ADNc, las secuencias son sometidas a un nuevo proceso de comparación con el algoritmo BLAST contra secuencias de distintas cepas de dicha bacteria. Después, en el paso siguiente, las secuencias son sometidas a un exhaustivo análisis en lo que podríamos llamar el centro de acopio o núcleo del flujo de trabajo. Esta herramienta se llama *StackPack* (www.egenetics.com/stackpack) dentro de la cual se incluyen las otras herramientas de ensamble y agrupamiento como son PHRAP, d2_cluster y CRAW, respectivamente. En este paso, las distintas secuencias se filtran para descartar las que en los pasos anteriores no han arrojado buenos resultados, es decir, que de esta manera, *StackPack* descarta aquellas secuencias contaminadas o defectuosas durante el proceso de secuenciación, además de llevar a cabo el ensamblaje final de las secuencias *EST* definitivas en sus respectivos *clusters*. Finalmente, se impone un filtro a los resultados parciales donde la longitud mínima de *ESTs* aceptadas será de 100 pb (pares de base) y donde se espera que tenga una asignación mínima de un 3%.

Para el caso particular de las pruebas llevadas a cabo con el flujo de trabajo del CIAT, éste arrojó como resultado final un total de 69 secuencias aceptadas a partir de una entrada de 300 secuencias, lo que evidenció la posible contaminación de las secuencias y la poca significancia de algunas de las obtenidas. Estos resultados demostraron la importancia de este flujo de trabajo para el análisis de *ESTs* en la búsqueda de genes de resistencia para *manihot esculenta* y el valor agregado por la funcionalidad y confiabilidad obtenida de la plataforma de análisis implementada. La ejecución coordinada del flujo de trabajo utilizando como motor a GPIPE, demostró la flexibilidad requerida en los flujos de trabajo bioinformáticos para la ejecución de este tipo de análisis y versatilidad al poder *instanciar* valores en los parámetros de cada uno de los métodos que componían el flujo de trabajo. Se obtuvo además la extensión funcional de la plataforma debido a la posibilidad de integración al sistema LIMS y la capacidad para almacenar los resultados y sus anotaciones de manera privada (o publica si así se desea) por parte de cada usuario.

4. Conclusiones

Se iniciará la construcción de componentes funcionales adicionales a la plataforma en todas sus capas: en el componente de fuentes de datos y almacenamiento se tiene un prototipo de diseño de un *Datamart* (bodega de datos de un dominio particular) para hacer análisis sobre los resultados de las ejecuciones de la plataforma. En su componente de flujos de trabajo, se iniciará la construcción de una capa de servicios *Web* que utilice BPEL4WS (www.ebpml.org/bpel4ws) como lenguaje de orquestación de servicios de forma que se pueda implementar una interacción de la plataforma con otras iniciativas que utilicen este mecanismo de integración de información.

Con respecto a la visualización, se espera iniciar un proyecto para la generación automática de la interfaz de la plataforma utilizando *portlets* como mecanismos de visualización, definidos a partir de servicios externos de ontologías y almacenados en un contenedor de *portlets* siguiendo la especificación JSR-168 (http://www.jcp.org/en/jsr/detail?id=168) que le permita ser portable entre diferentes plataformas.

5. Agradecimientos

Queremos expresar nuestros agradecimientos a Alex García, Samuel Thoraval, David Pinzón y Catherine Letondal por su continuo soporte con GPIPE y a los funcionarios del CIAT, Leonardo M. Galindo, Mauricio Soto, Fausto Rodríguez, Morgan Echeverri, Diana Bernal, Andrés Salcedo, Fernando Rojas, y Joe Tohme por su compromiso y disponibilidad a compartir sus conocimientos en el área biológica.

6. Referencias bibliográficas

Addis, M., Ferris, J., Greenwood, M., Li, P., Marvin, D., Oinn, T., & Wipat, A. (2003). *Experiences with e-Science workflow specification and enactment in bioinformatics*. In Proceedings of the UK e-Science All Hands Meeting 2003, Nottingham, United Kingdom, p. 459-466. http://eprints.ecs.soton.ac.uk/8991/

García-Castro, A. (2004) Implementación de servicios de análisis usando tecnología Open-Source. *Lecturas Matemáticas* 25 (2), 211-218. http://www.scm.org.co/revistas.php?modulo=Resultados&volumen=25&revista=lecturas

García-Castro, A., Thoraval, S., García, L. J., & Ragan, M.A. (2005). Workflows in bioinformatics: meta-analysis and prototype implementation of a workflow generator. *BMC Bioinformatics* 6:87. http://www.biomedcentral.com/1471-2105/6/87

Leo, P., Marinelli, C., Pappadà, G., Scioscia, G., & Zanchetta, L. (2004). *BioWBI: an integrated tool for building and executing bioinformatic analysis workflows*. Bioinformatics Italian Society Meeting (BITS 2004), Padova, Italy. http://bioinformatics.cribi.unipd.it/bits2004/abstracts/25.pdf

Ouzounis, C., & Valencia, A. (2003). Early bioinformatics: the birth of a discipline–a personal view. *Bioinformatics* 19 (17), 2176–2190. http://bioinformatics.oxfordjournals.org/cgi/screenpdf/19/17/2176

Soto, M., López, C., Restrepo, S., Piegu, B., Cooke, R., Delseny, M., Tohme, J., & Verdier, V. *Análisis de expresión global en la respuesta de defensa de la yuca a Xanthomonas axonopodis pv. manihotis*. http://www.redbio.org/rdominicana/redbio2004rd/Memoria_REDBIO_2004/simposios-PDF/s09-PDF/s09-03.pdf

Stevens, R. D., Robinson, S., & Goble, C. (2003). MyGrid: Personalised bioinformatics on the information grid. *Bioinformatics* 19 (Suppl.1), 302-304. http://bioinformatics.oxfordjournals.org/cgi/screenpdf/19/suppl_1/i302

Vailaya, A., Bluvas, P., Kincaid, R., Kuchinsky, A., Creech, M., & Adler, A. (2004) *An architecture for biological information extraction and representation*. In Proceedings of the 2004 ACM Symposium on Applied Computing, p. 103-110. http://bioinformatics.oxfordjournalds.org/cgi/reprint/bti187v1

Wilkinson, M. D., & Links, M. (2002) BioMOBY: an open source biological web services proposal. *Briefings in Bioinformatics* 3 (4), 331-341. http://bioinformatics.oxfordjournalds.org/cgi/reprint/3/4/331