PEDECIBA Informática
Instituto de Computación – Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay

Reporte Técnico RT 10-11

Taverna: un ambiente para el desarrollo de experimentos científicos

Guzmán Llambías **Raúl Ruggia**

2010

Taverna: un ambiente para el desarrollo de experimentos científicos
Llambías, Guzmán; Ruggia, Raúl
ISSN 0797-6410
Reporte Técnico RT 10-11
PEDECIBA
Instituto de Computación – Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay, 2010

Taverna: un ambiente para el desarrollo experimentos científicos

Guzmán Llambías, Raúl Ruggia
Instituto de Computación, Facultad de Ingeniería, Universidad de la República
Montevideo, Uruguay
{gllambi, ruggia}@fing.edu.uy

Resumen – Taverna es una potente herramienta para el diseño, desarrollo y ejecución de experimentos biológicos y componente central de la plataforma myGrid. Una de sus cualidades más importantes, es su capacidad de mediación entre usuarios y la computadora, simplificando de forma notable varias tareas computacionales avanzadas. En este reporte, se presenta el resultado de un relevamiento de la literatura, con el propósito de describir las principales características y funcionalidades de Taverna, así como la arquitectura, integración con otros proyectos, presentando también algunas limitaciones y debilidades.

Palabras clave: *Taverna, Middleware, Workflows*

I. INTRODUCCIÓN

Un experimento in-silico es un procedimiento que involucra recursos locales o remotos basados en la computadora para el desarrollo de pruebas que permitan verificar ciertas hipótesis, derivar resultados, buscar patrones o demostrar hechos conocidos [1]. En bioinformática, estos recursos pueden ser repositorios de información como las bases de datos EBML [4] y Swiss-Prot [5] o herramientas computacionales analíticas como ClustalW [5] y BLAST [7]. Los bioinformáticos desarrollan sus experimentos in-silico siguiendo un esquema de workflow, combinando diferentes recursos cuidadosamente ordenados donde cada recurso procesa ciertos datos y genera resultados. Por ejemplo, un workflow para investigar la evolución de las relaciones entre proteínas puede comenzar con adquirir una secuencia de aminoácidos del repositorio Swiss-Prot y luego aplicar el algoritmo de ClustalW para alinear e identificar patrones entre las secuencias.

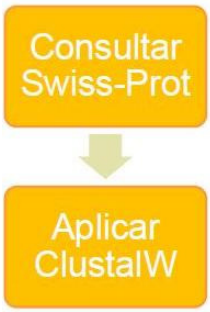

Figura 1 - Esquema de un workflow científico

Originalmente, la implementación de estos "workflows científicos" tenía características rudimentarias, llegando a utilizarse mecanismos "cut & paste" entre aplicaciones Web o scripts de bajo nivel basados en lenguajes como Python, para su consulta automática y el procesamiento de resultados. Hoy día, la situación ha cambiado y muchas de las grandes bases de datos biológicos, así como principales herramientas analíticas, son expuestas vía Web Services como forma de garantizar un acceso estándar [8]. Por otro lado, han surgido varias herramientas y lenguajes de programación que permiten diseñar y ejecutar los workflows científicos orquestando Web Services. En ese sentido, el proyecto myGrid [9], un proyecto de middleware en e-Science, ha desarrollado herramientas y aplicaciones basadas en middleware de muy alta abstracción, siendo Taverna, una de ellas [1][2].

Taverna permite la automatización de experimentos vía el uso de diferentes servicios de diversa índole, que van desde biología, química y medicina, hasta música, meteorología y ciencias sociales. Una de las características más interesantes de Taverna, es que permite que científicos con poco conocimiento técnico en computación, sean capaces de desarrollar sus trabajos en base al uso de herramientas con capacidades computacionales avanzadas y complejas, como puede ser el acceso a un Web Services, el descubrimiento de recursos, el registro de Provenance, manipulación de datos y XML entre otros.

En este reporte, se presenta el resultado de un relevamiento de la literatura en torno a Taverna, con el propósito de describir brevemente sus principales características y funcionalidades. Es así, que en la siguiente sección se presenta el proyecto myGrid y su relación con Taverna. En la sección III, se presenta la arquitectura de Taverna, así como sus principales funcionalidades nativas y foráneas, producto de la integración con terceros. En la sección IV, se presentan algunas limitaciones y debilidades. Por último, en la sección V, se presentan las conclusiones del trabajo.

II. PROYECTO myGRID

myGrid es un proyecto de middleware en e-science iniciado en el año 2001, con el propósito de proveer herramientas de middleware de muy alta abstracción, que simplifiquen a los biólogos y bioinformáticos, el desarrollo de sus experimentos. Algunas de las herramientas desarrolladas son:

- Taverna: para el diseño, desarrollo y ejecución de experimentos
- myExperiment: para la colaboración entre grupos científicos, permitiendo la publicación, curación y búsqueda de experimentos (workflows).
- BioCatalogue y Feta: para la publicación, categorización y búsqueda de servicios Web
- Utopia: para una visualización y análisis de los resultados.

Todos estos componentes fueron desarrollados en base a tecnologías de Web Services, Workflows, Web 2.0 y Semánticas, siguiendo una filosofía de código abierto y alta extensibilidad por parte de terceros. Cada componente fue diseñado para trabajar en conjunto siguiendo el paradigma SOA, pero teniendo en cuenta un uso independiente y posibles extensiones para propósitos específicos. El uso conjunto de éstos permite conformar la plataforma base para el desarrollo de entornos virtuales de investigación científica, denominados **e-Labs** [34].

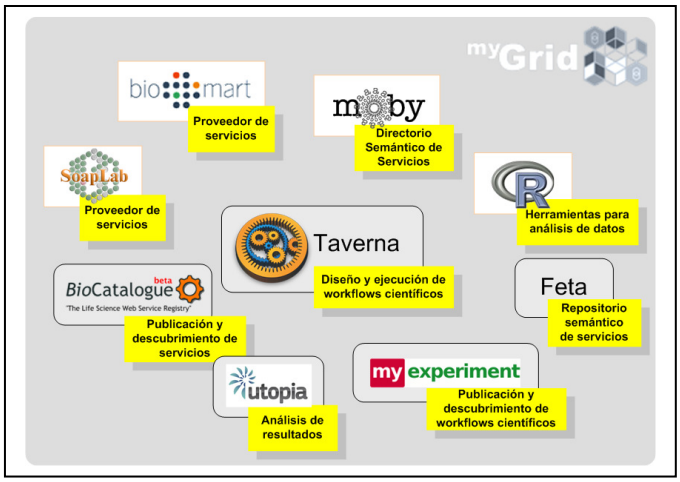

Figura 2 - Componentes de la plataforma myGrid

La Figura 2 ilustra los principales componentes de la plataforma myGrid, al igual que algunas extensiones desarrolladas por terceros que enriquecen la plataforma con novedosas funcionalidades. En particular, los proyectos BioMoby y R proveen un repositorio semántico de servicios y herramientas para el análisis de datos, mientras que los proyectos BioMart y SoapLab, proveen a la plataforma Servicios Web específicos para el dominio biológico. En la figura, se distinguen a las extensiones con el color blanco y con gris a los componentes myGrid.

Por otro lado, la Figura 3 presenta una vista lógica del diseño de la plataforma, ubicando a cada componente en una capa de acuerdo a características, funcionalidades y relaciones que existen entre ellos. En este caso, se pueden apreciar tres capas: presentación, aplicación y servicios. En la capa de presentación se ubican los componentes de cada herramienta encargados de proveer una interfaz de usuario para su uso. En la capa de aplicación, se presentan los componentes de cada herramienta encargados de la ejecución del negocio. Por último, en la capa de servicios se encuentran los distintos proveedores de servicios que permiten el diseño y ejecución de un workflow. Nuevamente, se presentan en blanco los componentes desarrollados por terceros y en gris los componentes propios de la plataforma myGrid.

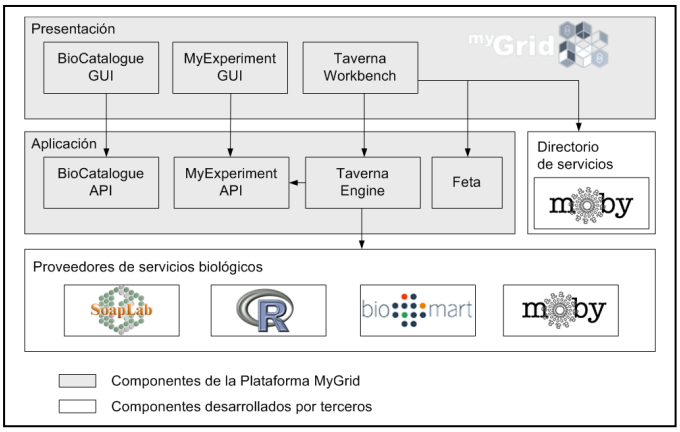

Figura 3 - Vista lógica de la plataforma myGrid

III. TAVERNA: CORAZÓN DE LA PLATAFORMA MYGRID

Taverna es el componente central de la plataforma myGrid, el cuál engloba un conjunto de componentes para el diseño, desarrollo y ejecución de "workflows científicos". Taverna está compuesto por:

- Taverna Engine: un motor de workflows.
- Taverna Workbench: un cliente de escritorio para el diseño de workflows
- Taverna Server: un servidor para la ejecución de workflows remotos.

Actualmente, Taverna se encuentra en la versión 2.1.2 e incluye a Taverna Engine y Taverna Workbench. Taverna Server se encuentra aún en desarrollo en una versión alfa y por ahora, se desaconseja su uso.

A continuación se presenta la arquitectura interna de Taverna, el modelo de ejecución de workflows, sus principales características nativas y foráneas (producto de la integración con terceros), así como otras características clasificadas como "de interés".

A. Arquitectura

La Figura 4 presenta la arquitectura interna de Taverna, identificando cada uno de sus componentes y las relaciones que existen entre ellos. En esta figura, se presenta únicamente los componentes relativos al Workbench y Engine, dejando ausentes los relativos al Servidor Remoto.

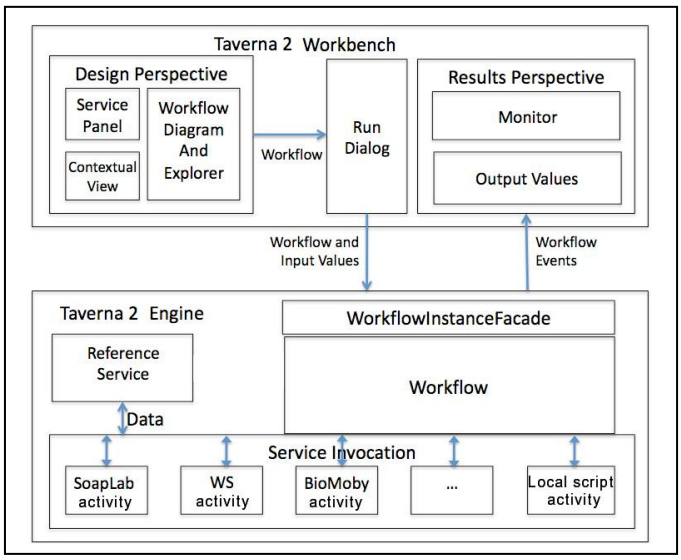

Figura 4 - Arquitectura de Taverna.

B. Modelo de ejecución de workflows en Taverna

En Taverna, los workflows son considerados grafos de *procesadores*, donde cada procesador recibe en sus *input ports* un conjunto de datos de entrada y produce un conjunto de datos salida, los cuales retorna en sus *output ports*. Cada arco del grafo se denomina *data link* y conecta un par de puertos (output con input) especificando una dependencia entre las salidas de un procesador (*source*) y las entradas de otro (*sink*). Los tipos de datos permitidos por el modelo son tipos de datos simples (string, integer, boolean, etc.) y listas de tipos de datos simples con niveles de anidamiento arbitrario.

Conceptualmente, la ejecución de un workflow comienza con la inyección de datos al mismo utilizando los *workflow inputs* (datos de entradas del workflows). Estos datos serán redirigidos por los data links a cada uno de los procesadores sucesores, los cuales comenzarán la ejecución una vez que todos sus *input ports* son poblados con datos. La ejecución del procesador implica la ejecución de una actividad que puede ser la invocación a un Web Service, la ejecución de un script Java (Beanshell script) o cualquier otro tipo de procesador definido en Taverna. Los procesadores nativos a Taverna son: Arbitrary WSDL (Web Services tradicionales), SoapLab (comandos de consola encapsulados en Web Services) [10], Talisman, Nested Workflow, String constant y BeanShell Processor (scripts Java). Una vez finalizada la actividad del procesador, éste envía a sus *output ports* los resultados de la ejecución.

La Figura 5 presenta la estructura interna de un procesador Taverna, el cuál presenta un diseño basado en el patrón *Interceptor* y en la definición de un *dispatch stack*. Específicamente, se define un conjunto de capas asociado a cada procesador, donde cada capa se encarga de una tarea específica previo el envío de la solicitud. Por defecto, la configuración de los procesadores está compuesta por las siguientes capas:

- **Parallelise:** en casos donde los parámetros de entrada incluyan un elemento de tipo lista, esta capa se encarga de ejecutar en forma paralela un procesador por cada elemento de la lista.
- **Error Bounce:** esta capa es responsable de terminar la ejecución del workflow cuando exista algún error en las entradas de algún procesador.
- **Failover:** esta capa se encarga de detectar fallas en la ejecución del procesador y es la responsable de seleccionar una actividad alternativa.
- **Retry:** esta capa es la encargada de proveer tolerancia a fallas a los workflows. En caso de error en la ejecución de una actividad, se re ejecutará un número configurable de veces antes de producir un error.
- **Invoke:** esta capa es la encargada de realizar la invocación a la actividad específica. Puede ser la invocación a un Web Service, un script Java o cualquier otra actividad definida en Taverna.

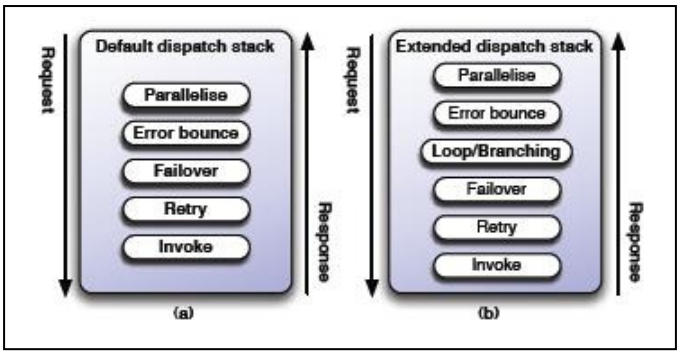

Figura 5 - Estructura interna del procesador Taverna

Una vez recibida la respuesta a la solicitud, el mensaje realiza el camino inverso pasando por cada una de las capas definidas hasta ser enviado al *output port y data link* correspondientes.

Este esquema de capas puede ser extendido permitiendo nuevos comportamientos por los procesadores. Actualmente, se están desarrollando dos extensiones que incluyen una capa *While-Loop* y una capa *Provenance*.

Como se puede observar, los workflows en Taverna siguen un modelo de ejecución orientado a datos y no orientados a control como pueden ser los procesos WS-BPEL u otros tipos de workflows empresariales. Esto se debe principalmente, a la diferencia de requerimientos entre el mundo empresarial y el mundo biológico. El primero, requiere un control de cómo se realizan las actividades, mientras que el segundo, se enfoca en el resultado de las operaciones, sin importar el orden de ejecución. En [11] se realiza una interesante comparación entre Taverna y WS-BPEL, mencionando algunas de estas diferencias.

A continuación se describe un ejemplo de un experimento in-silico dentro del contexto biológico implementado en Taverna.

C. Ejemplo de un workflow científico

La obtención de información sobre el metabolismo o las vías de señalización (pathways) asociadas a un gen es un escenario común de análisis dentro del contexto de bioinformática [35]. Una forma de desarrollar este análisis, es mediante la integración de datos presentes en las bases de datos de secuencias de nucleótidos con los datos presentes en las bases de datos sobre vías de señalización. La Figura 6, muestra un workflow desarrollado en Taverna que realiza un análisis de este tipo.

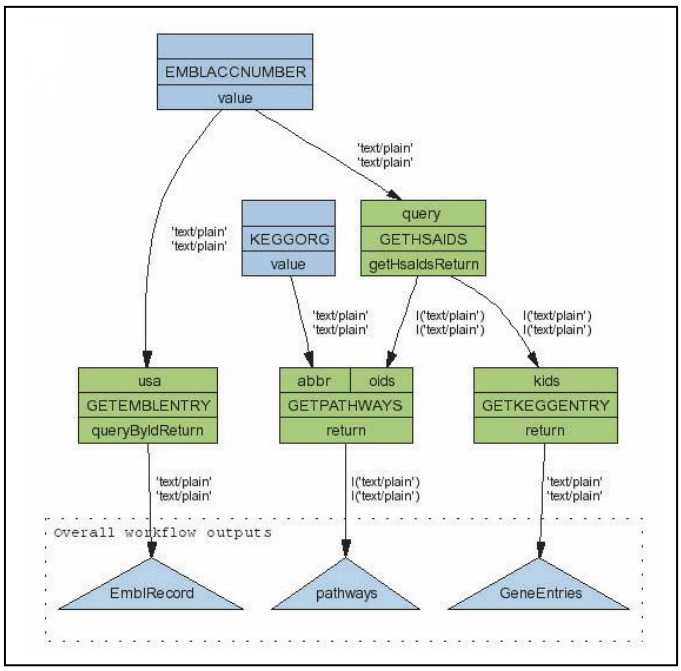

Figura 6 - Ejemplo de workflow implementado con Taverna

El workflow recibe en sus *input ports* una lista de números EMBL de acceso a un gen y retorna en sus *output ports* tres datos:

1. EmblRecord: un registro EMBL asociado al número EMBL de acceso a un gen proveniente del *input port*
2. Pathways: una lista con las vías de señalización (pathways) asociadas al número EMBL de acceso a un gen.
3. GeneEntries: una lista con los genes asociados a un número EMBL de acceso a un gen.

Para obtener el EmblRecord, el workflow simplemente consulta el servicio KEGG (método GetEmblEntry) en busca del registro apropiado, utilizando los datos provistos en el *input port* del workflow.

La obtención de los Pathways no es tan directa y requiere cierto procesamiento extra. En primer lugar, es necesario consultar el servicio KEGG (método GetHSAIDs) para obtener los identificadores KEGG del gen asociado al número EMBL (del *input port* del workflow). Con este identificador, es posible consultar la base de datos de pathways utilizando el servicio KEGG (método GetPathways) y así obtener todos los Pathways del gen asociado.

Por último, para obtener los GeneEntries se consulta el servicio KEGG (método GetKEGGEntry) utilizando el identificador del gen. Este identificador es el mismo que se utiliza para consultar la base de datos de pathways.

D. Características nativas

En esta sección se presentan algunas de las características nativas de Taverna. Se entiende por nativas, a todas aquellas características que fueron desarrollados por el equipo Taverna, sin participación de terceros.

1) Iteraciones implícitas

En Taverna, cada parámetro de entrada de un procesador tiene un tipo de dato definido, el cual puede ser simple (string, int, etc.) o una lista, posiblemente anidada, de tipos simples. Es así, que puede ocurrir que durante la ejecución de un workflow, una de las entradas del procesador no se corresponda totalmente con el tipo declarado. Por ejemplo, un procesador con un parámetro de salida de tipo "lista de strings", podría ser conectado a un procesador con entrada "string". En estos casos, Taverna presenta un comportamiento interesante, interpretando esta diferencia de tipos como un indicador para ejecutar el procesador de forma iterativa. Esto es, ejecutar el procesador una vez por cada elemento de la lista. Por cada diferencia en el nivel de anidamiento entre el valor actual de entrada y el tipo de entrada del procesador, Taverna disparará automáticamente una iteración implícita del mismo, de acuerdo a la cantidad de elementos de la lista. Vale la pena mencionar, que ante diferencias que van más allá del nivel de anidamiento, como puede ser un string en lugar de int, Taverna lanzará un error.

Al momento de diseñar un workflow, puede ocurrir que se utilicen procesadores con dos o más parámetros de entrada de tipo lista. En estos escenarios, Taverna ofrece dos opciones de comportamiento: 1) realizar el producto cartesiano o 2) el producto escalar de las listas. Este último caso únicamente es posible, si el tamaño de las listas involucradas es la misma. Es importante resaltar que Taverna permite ejecutar todas las iteraciones de forma concurrente, siempre y cuando los recursos computacionales sean suficientes (ver capa Parallelize de la sección B).

El soporte a iteraciones implícitas ofrecido por Taverna es sumamente importante, dado que este patrón de comportamiento ocurre con frecuencia en el desarrollo de workflows científicos. Desde el punto de vista del usuario, Taverna ofrece una potente abstracción que impacta de forma sustancial en el diseño, simplificándolo notablemente. Los usuarios no deben preocuparse de este tipo de iteraciones, ya que éstas serán llevadas a cabo de forma transparente por el motor de ejecución.

2) Provenance

De acuerdo al diccionario Cambridge de la lengua Inglesa, Provenance es el lugar de origen de algo. En un contexto de workflows científicos, Provenance, es la traza de todas las transformaciones de datos ocurridas durante una ejecución y puede ser utilizada para guiar el análisis *post mortem* de un experimento. Por ejemplo, pueden ser utilizadas para responder preguntas acerca del linaje de los datos, esto es, qué entradas y qué transformaciones, dieron lugar a los resultados obtenidos en una ejecución dada. Continuando con el ejemplo presentado anteriormente, una de las preguntas que se podrían contestar con las trazas de Provenance, sería:

- ¿Por qué el workflow dio determinado pathway como dato de salida?

En otras palabras:

- ¿Qué elemento de la lista de números EMBL está asociada a determinado elemento de la lista de pathways resultante?

Dada la importancia de la información de Provenance, Taverna fue desarrollado con el propósito de permitir su captura y posterior análisis. En ese sentido, Taverna no sólo permite identificar qué entradas, dieron lugar a qué salidas en un procesador dado, si no, que también permite determinar qué entradas y procesamientos del workflow dieron lugar a qué

dato de la salida. Un análisis como este puede ser realizado utilizando el Workbench o aplicaciones externas que realicen consultas al almacén de Provenance de Taverna. Dicho almacén se encuentra implementado mediante una base de datos relacional y puede ser consultada utilizando consultas SQL o de forma más específica y en alto nivel, mediante consultas a una API especializada en Provenance. Esta última, puede retornar los datos de dos formatos: 1) nativo a Taverna o 2) en formato RDF compatible con *Janus*, una ontología para describir información de Provenance [33].

Actualmente, el Workbench Taverna provee soporte a la captura y análisis de Provenance, que permite responder qué entradas y qué procesadores dieron lugar a qué salida de datos. Sin embargo, aún no existe soporte desde el Workbench para responder consultas más "finas" que permitan determinar qué dato de una lista de entradas y qué procesadores, dieron lugar a una salida específica del workflow. Dicho soporte se encuentra aún en etapa de desarrollo, pero se espera su incorporación a la brevedad.

E. Integración con terceros

En esta sección, se describen algunas de las características foráneas más importantes de Taverna, producto de la integración con terceros.

1) Integración con myExperiment

Una de las grandes virtudes de los workflows para el modelado de experimentos in-silico, es su capacidad de facilitar el trabajo manual y repetitivo. Sin embargo, esta no es la única, ni la más importante, ya que estos tienen una característica extremadamente útil para el dominio biológico: ser reutilizables.

Los científicos pueden reutilizar un workflow con diferentes datos, pueden modificarlo como parte de su trabajo diario o pueden compartirlo con sus colegas como forma de publicación. Una prueba de ello, fue la identificación de genes relacionados con la tolerancia a la trypanosomiasis en una región de África [22], donde la modificación de datos de un workflow existente dio lugar a nuevos descubrimientos en el área. En esta ocasión no fueron los datos producidos por el workflow inicial lo que permitieron los nuevos descubrimientos, si no, su reutilización con nuevos datos. Esto es, el reuso del *know-how* de los procesos y procedimientos científicos, el cual muchas veces se encuentra implícito u oculto. A partir de este tipo de experiencias, los workflows pasan a ser potenciales activos, valorados como artefactos de primera clase que pueden ser compartidos, comerciados y reutilizados entre las comunidades científicas.

Sin embargo, previo a su reutilización es necesario descubrirlo y para ello se han identificado un conjunto de propiedades que permiten guiar su descubrimiento. Algunas de ellas son:

- una descripción funcional y propósito del workflow;
- documentación de servicios utilizados, con ejemplos de datos de entrada y salida;
- información de Provenance;
- reputación, dueño y permisos de uso;
- información de calidad;
- dependencias con otros workflows.

Asociando estas propiedades a los workflows, es posible asistir a los científicos en su descubrimiento, reuso y nuevos propósitos, dando una interpretación precisa de los mismos [21]. En ese sentido, el proyecto myGrid ha desarrollado myExperiment, un ambiente virtual de investigación (Virtual Research Environment) para compartir workflows. Lo más destacado de myExperiment es que ha sido muy exitoso en la curación de workflows, ya que gracias a su enfoque Web 2.0 y social, los propios usuarios han ido etiquetando y verificando la calidad y estado de los mismos, dando lugar a que en myExperiment, los workflows y servicios son creados y mantenidos por la propia comunidad de usuarios.

Algunas de las características más importantes de myExperiment son su capacidad de ejecución en modo federado, así como una API de funcionalidades básicas para consulta, publicación y anotación de workflows. Está última, fue utilizada para el desarrollo de un plug-in para Taverna, que permite la integración de myExperiment al Workbench.

Actualmente, dicho plug-in permite realizar búsquedas en myExperiment, examinar los workflows existentes, visualizar su nube de etiquetas, descubrir workflows basándose en estas, previsualizar workflows, importar workflows hacia Taverna y por último, guiar al usuario a un workflow dentro del sitio.

Utilizando Taverna y myExperiment, el ciclo de vida de un experimento se puede componer de las siguientes etapas ilustradas en la Figura 7:

1) los científicos utilizan myExperiment para descubrir un workflow de uso público utilizando algunas de sus técnicas de descubrimiento (búsqueda, etiquetado, navegación, etc.);
2) el científico descarga/importa el workflow en Taverna y lo edita;
3) el científico sube una nueva versión del workflow a myExperiment.

Uno de los beneficios del plug-in, es la edición automática de cierta información que debe ser ingresada por el científico durante la etapa de publicación. El plug-in editará automáticamente el nombre del workflow, sus entradas, salidas, servicios que utiliza y dependencias con otros workflows.

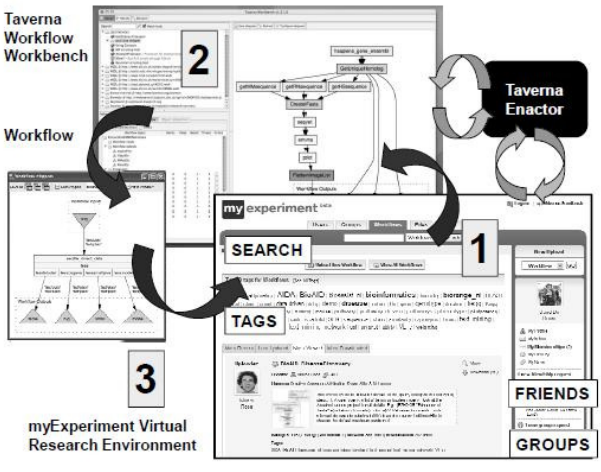

Figura 7 - Ciclo de vida de un workflow utilizando myExperiment

Actualmente, el equipo myExperiment está enfocado en cómo asociar otros tipos de información a los workflows y así desarrollar *Research Objetcts* [23]. Los Research Objects son objetos que agrupan un conjunto diverso de contenido digital de una investigación, como ser: personas, estudios, metodologías, materiales y resultados. El objetivo principal de los Research objects es el intercambio y reuso de información entre e-Labs. El uso de Research Objects, aún no tiene soporte dentro del plug-in Taverna.

2) Integración con el lenguaje R

La era post-genómica trajo consigo grandes cambios en la biología, incrementando la cantidad de datos que deben ser a analizados por los bioinformáticos. En particular, los análisis genómicos del transcriptoma, proteoma y metaboloma, cuyas medidas son realizadas en paralelo utilizando tecnología de alto rendimiento con técnicas de microarrays y espectrometría en masa. Estos análisis se basan en la performance de los experimentos in-silico y la detección de patrones en busca de similitudes fenotípicas. A su vez, se realizan pruebas estadísticas y se relacionan sus resultados con información alojada en bases de datos biológicas, para así obtener alguna conclusión. Por ejemplo, el análisis de la expresión genética generada de los experimentos de microarrays, consiste en el siguiente conjunto de pasos:

1. normalización y estandarización de los datos.
2. un análisis estadístico
3. interpretación de los datos vía la anotación de genes con información relacionada a su función biológica [25].

Esta sección se centra en el segundo paso, el análisis estadístico de los datos, en donde se presentará una introducción a R y su integración al Workbench Taverna.

R es un lenguaje y un entorno de programación para el desarrollo de análisis estadísticos y gráficos. Algunas de sus características más importantes son el manejo efectivo de datos y su almacenado; un conjunto de operadores para realizar cálculos sobre arreglos y matrices; una extensa e integrada colección de herramientas para análisis de datos; facilidades gráficas para visualizarlos; un lenguaje de programación simple y efectivo que incluye, condicionales, ciclos, funciones recursivas y soporte a entrada y salida de datos de tipo I/O.

Dada la popularidad de R dentro de la comunidad biológica y bioinformática, fue desarrollado un plug-in Taverna para utilizar R en el diseño de workflows. El nuevo plug-in provee un nuevo procesador, llamado **RShell**, que permite desarrollar scripts en lenguaje R, de forma similar a como el procesador BeanShell permite desarrollar scripts en Java. De esta forma, los científicos pueden integrar información del workflow (resultados de consultas a bases de datos o ejecución de herramientas analíticas) a los scripts R de una forma sencilla y sin la necesidad de una integración manual. Vale la pena resaltar que a diferencia del procesador BeanShell, los scripts generados con RShell no son ejecutados localmente por el motor Taverna, si no, en un servidor R. Al momento de ejecución, el procesador RShell actúa de cliente, el cual se comunica con un servidor R vía TCP/IP utilizando la biblioteca RServe [27]. El servidor ejecuta el script y retorna el resultado a cliente, el cual lo redirige al Workbench. La Figura 8, presenta de forma gráfica la integración de Taverna con R, RShell y RServe.

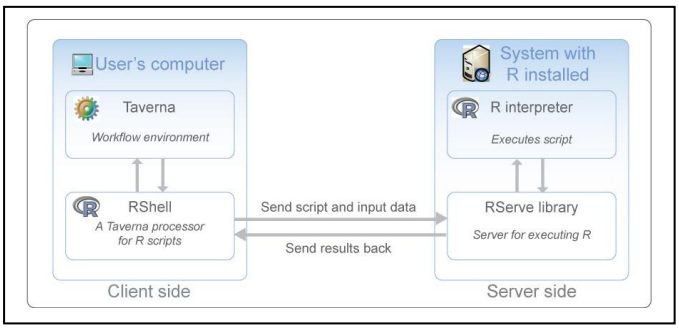

Figura 8 - Integración con el lenguaje R

Actualmente, el plug-in R provee un soporte para la ejecución de scripts R versión 2.7 o superior, soporte a parámetros de salida en formato PDF y PNG, edición de código con sintaxis resaltada, envío y recepción de archivos de texto desde RShell al servidor R, múltiples parámetros de salida, soporte a tipos de dato integer, double, string, boolean y R-expressions, carga de scripts desde archivos y manejo de sesiones R [28].

A pesar que la integración de R al Workbench Taverna es sumamente importante, existen ciertas limitaciones que dificultan el desarrollo de workflows con R. Un ejemplo de ello es el análisis de datos de microarrays presentado en [25], el cual requiere de un conocimiento del lenguaje R (para el desarrollo de análisis estadísticos) y experiencia en Java (para el desarrollo de scripts para el re formato de datos e interfaces de usuario de consulta). La falta de experiencia en estos dos lenguajes puede llegar a dificultar o hasta prohibir que usuarios poco avanzados, construyan workflows con un complejo componente de análisis de datos.

3) Integración BioMoby

BioMOBY es un proyecto de investigación Open Source con el objetivo de explorar nuevas formas de implementar registros de Web Services que faciliten el descubrimiento y distribución de datos biológicos. Su propuesta consiste en extender el actual paradigma de Web Services e implementar un modelo de registro de servicios que permita una búsqueda y recuperación basada en jerarquías de objetos y servicios.

En BioMOBY existen tres componentes básicos: los Objetos MOBY, la Central MOBY y las jerarquías MOBY (una de objetos y otra de servicios). Los Objetos MOBY son datos estructurados que se transfieren entre clientes y servicios; la Central MOBY, es un registro que para cada servicio, almacena su URL, el tipo de servicio y los tipos de objeto de entrada y salida que este tiene. Por último, las jerarquías describen las distintas relaciones que existen entre objetos y servicios. Ambas jerarquías se encuentran definidas en RDF [15], [16].

Como se mencionó anteriormente, el principal objetivo del proyecto BioMoby es explorar nuevas formas de registro de servicios que potencien su búsqueda y recuperación. En ese sentido, los clientes BioMOBY pueden consultar la Central en búsqueda de servicios de acuerdo a los tipos de objetos de entrada, los tipos de objetos de salida, el tipo de servicio, la autoridad o cualquier combinación de estos. Con estos datos y las jerarquías Moby, la Central emite una lista de autoridades con los servicios que éstas proveen y según las características requeridas. Uno de los clientes Moby para llevar a cabo este tipo de consultas, es el plug-in BioMoby para Taverna.

El plug-in BioMoby, permite realizar consultas "avanzadas" de servicios sobre la Central, basándose en los tipos de entrada. A su vez, ofrece una asistencia inteligente a los usuarios, garantizando propiedades de tipo *type safety,* así como guías para el diseño del workflow. El editor guía al usuario en el correcto uso de los tipos de datos y permite a partir de los resultados de las búsquedas, ir enlazando de forma automática los diferentes servicios. Por último, el plug-in ofrece constructores y *parsers* que permiten ensamblar el mundo de los servicios BioMOBY, con el mundo de los servicios tradicionales, permitiendo aprovechar las características de ambos.

Desafortunadamente, el plug-in aún no cuenta con todas las capacidades que BioMoby ofrece en solitario, limitando las búsquedas de servicios únicamente a los tipos de datos de entrada. Hasta el momento, no permite realizar búsquedas de acuerdo al tipo de servicio, a los tipos de salida, ni palabras clave.

A pesar de estas limitaciones, el aporte BioMoby es sustancial ya que incorpora características innovadoras para el desarrollo y diseño de workflows con Taverna. Hasta el momento, Taverna no poseía un buscador de servicios con tales características, limitándose a consultas de tipo sintácticas basadas en los nombres de los servicios. Con la incorporación de BioMoby, es posible realizar búsquedas semánticas que posibiliten mejores resultados. A su vez, vale la pena resaltar sus cualidades de asistencia al usuario, ya que hasta el momento no se conoce asistencia similar en otras herramientas de workflow open source, incluyendo las herramientas de workflows empresariales como Eclipse BPEL o jbpm Designer.

4) Integración con SADI

SADI o Semantic Automated Discovery and Integration es la evolución del proyecto BioMoby y plantea nuevas técnicas para el desarrollo de registros de Web Services, su categorización y posterior búsqueda y recuperación [36].

SADI no propone nuevos estándares, tecnologías, formatos de mensajes o estructuras de datos. Simplemente, ordena los estándares existentes y sugiere el uso de buenas prácticas para la representación de datos y el intercambio de mensajes utilizando las tecnologías tradicionales de los Web Services (XMLSchema, SOAP, WSDL) y la Web Semántica (OWL, RDF, SPARQL).

Las bases de su propuesta se basan en la observación del comportamiento de los Web Services del área bioinformática. Con algunas raras excepciones, estos Web Services son independientes, idempotentes, sin estado, trasformativos y atómicos, presentando un contraste importante con respecto a los Web Services empresariales. Al ejecutar un Web Service, los usuarios finales no tienen en cuenta un proceso o modelo de negocio. Simplemente, están interesados en conocer la relación que existe entre los datos de entrada y los datos de salida. Por ejemplo, al ejecutar un proceso BLAST, lo importante es la búsqueda de patrones en las secuencias y no, la ejecución de una matriz de similitud. Teniendo en cuenta estas características, SADI se basa en exponer de forma explícita la relación semántica que existe entre los datos de entrada y salida de un Web Service.

A diferencia de BioMoby u otros proyectos de clasificación semántica de servicios que se preocupan por definir los tipos de datos de entrada y salida, SADI pone foco en definir predicados que permitan describir la relación entre las entradas y las salidas de un servicio y a partir de estas propiedades, establecer una mejor clasificación de servicios y garantizar mejores niveles de búsqueda y recuperación.

Actualmente, existe un plug-in SADI para Taverna con características muy similares a las provistas por el plug-in BioMoby, pero utilizando los mecanismos de búsqueda descritos anteriormente que lo dotan de una mayor potencia y precisión. Asimismo, el plug-in mantiene las características de asistencia al usuario propuestas por BioMoby y las mejora, encontrándose avances desde el punto de vista visual, de navegación y una mayor información al usuario a la hora de seleccionar los servicios.

5) Integración con planillas Excel

Las planillas Excel han resultado ser una herramienta sumamente útil en diversas áreas de la sociedad, llegando a sustituir en determinados casos a los tradicionales sistemas contables. Lo que resulta interesante, es que este tipo de "micro sistemas" posee su propia base datos, sus propios servicios de procesamiento (formulas), generación de reportes y en determinados casos, integración con sistemas externos (MS Office provee integración con servicios WCF). Observando todas estas características, no es extraño que el equipo Taverna haya decidido incorporarlas en el diseño de los workflows [17].

Actualmente, Taverna ofrece la capacidad de incorporar planillas excel a los workflows de la misma forma que cualquier otra fuente de datos (WS, SoapLab, etc.). Esto es, vía el uso de procesadores. **SpreadsheetImport**, el nuevo procesador, ofrece la posibilidad importar/exportar datos de una planilla excel/workflow al workflow/planilla permitiendo seleccionar filas, columnas, administrar celdas vacías y etiquetado de datos de una columna. Por el momento, Taverna ofrece soporte únicamente a planillas Excel 2007/OOXML y OpenOffice/ODF.

Una de las limitaciones encontradas en el uso de las planillas, es que durante su importación, las fórmulas definidas no son recalculadas, por lo que no será posible utilizar dichos datos como valores de entrada.

A pesar de las limitaciones encontradas, el uso de planillas excel parece prometedor, permitiendo a los usuarios reutilizar datos legados definidos en planillas excel, ejecutar los experimentos y luego exportarlos en una nueva planilla, para su posterior procesamiento en reportes, nuevos cálculos, etc.

F. Otras características de interés

En esta sección se describen otras características interesantes relativas a Taverna, pero más relacionadas a infraestructura e integración de aplicaciones.

1) Integración con Web Services seguros

En los últimos años, grandes bases de datos biológicas así como diferentes herramientas analíticas fueron expuestas vía Web Services como forma de garantizar un acceso estándar a las mismas [8]. Por otro lado, la aparición de nuevos estándares para Web Services hizo viable el desarrollo de Web Services avanzados con características de seguridad, transacciones y mensajería confiable, lo que provocó que varias organizaciones biológicas, así como grupos de investigación comiencen a extender sus Web Services básicos con algunas de estas características. En particular, comienzan a surgir Web Services seguros basados en el estándar WS-Security.

WS-Security [30] es un estándar desarrollado por la OASIS que permite el desarrollo de Web Services seguros,

garantizando la integridad, confidencialidad y autenticidad de los mensajes SOAP. WS-Security provee una gran variedad de opciones para la autenticación de mensajes incluyendo soporte usuario y contraseña, certificados X.509, tickets Kerberos y tokens SAML. A su vez, incorpora la fecha de creación (timestamp) de los mensajes a los mensajes SOAP como forma de evitar ataques de replay.

Una de las principales ventajas de WS-Security frente a otros mecanismos como SSL e IPSec, es que este permite una seguridad a nivel de aplicación en lugar de una seguridad a nivel de transporte como estos últimos ofrecen.

Dadas las características de los nuevos Web Services biológicos, Taverna comienza a brindar soporte para algunas de posibilidades que brinda WS-Security. En particular, Taverna se enfoca en la autenticidad de mensajes vía el uso de usuario y contraseña, el cual puede ser enviado en texto plano o digest. Vale la pena recordar, que digest es un mecanismo de cifrado débil por lo que se desaconseja su uso como único mecanismo de protección de claves [31] y se recomienda reforzar la seguridad con un cifrado a nivel transporte como puede ser TLS/SSL o a nivel de mensajes como los provistos por WS-Secuirty. En ese sentido, Taverna aún no ofrece cifrado WS-Security pero si, soporte a SSL, por lo cual se puede utilizar autenticación WS-Security basada en usuario y contraseña, protegido con SSL [18], [19].

IV. Limitaciones y debilidades

Como se mencionó a lo largo del documento, Taverna cuenta con un gran número de características y funcionalidades muy potentes para llevar a cabo el diseño, desarrollo y ejecución de workflows científicos. Algunas de ellas, muy novedosas con respecto a herramientas similares aplicadas a contextos empresariales. Sin embargo y a pesar de todas estas bondades, Taverna sufre de las siguientes limitaciones a tener en cuenta.

A. Formato de datos

El formato de datos previo al consumo de los procesadores es una tarea básica y fundamental en el diseño y desarrollo del workflow. Dicha tarea implica adaptar el formato en que se encuentran los datos a un formato entendido por el procesador. Dada la gran cantidad de formatos existentes en el mundo biológico, esta tarea puede llegar a consumir una gran cantidad de horas, principalmente, por la poca documentación y la complejidad de los mismos. En ese sentido, Taverna ofrece poco soporte para el desarrollo de transformaciones, delegando esta tarea al uso de *Shim Services* basados generalmente en procesadores BeanShell. Sería deseable contar con un soporte más avanzado o "transformaciones especializadas" al contexto biológico que asistan al usuario, simplificándole esta tarea.

B. Sustitución de servicios

Cambios imprevistos en las interfaces de los servicios, caídas temporales y errores internos inesperados, son algunos de los problemas que pueden ocurrir durante la ejecución del workflow. Para resolver este problema, Taverna propone la ejecución de servicios alternativos en caso de ocurrir errores en un servicio "primario". Este soporte es efectivo pero manual, y sería deseable contar con prestaciones más inteligentes y automáticas, que se encargue de la reconfiguración del workflow, minimizando al máximo las intervenciones humanas.

C. Eventos

La programación orientada a eventos es un paradigma de programación donde el flujo de ejecución es guiado por eventos (p ej. sensores, mensajes, etc.). Este paradigma es típicamente utilizado en el desarrollo de interfaces de usuarios, pero su uso también es frecuente en el desarrollo de workflows empresariales, siendo muy común el evento de expiración de una tarea por vencimiento en el tiempo de espera. Lenguajes de especificación de workflows empresariales como WS-BPEL, Windows Workflow Foundation o JBPM cuentan con primitivas y constructores específicas para el manejo de eventos. Lamentablemente, Taverna no provee un soporte nativo con tales características. Actualmente, existen experimentos alojados en ^{my}Experiment se beneficiarían de contar con un soporte a eventos [37].

D. Asincronismo

En un escenario cliente-servidor, la comunicación asincrónica es un mecanismo sumamente útil si el servidor debe consumir grandes cantidades de tiempo para llevar a cabo la tarea solicitada, ya que los clientes son liberados para proseguir con otras actividades, en contraposición con una comunicación sincrónica, donde los clientes deben esperar "colgados" por su finalización. En ese sentido, la comunicación asincrónica resulta sumamente interesante dentro del contexto biológico, ya que los procesos analíticos comúnmente insumen una gran cantidad de tiempo. Lamentablemente, Taverna aún no cuenta con un soporte nativo a comunicaciones asincrónicas y estas deben desarrollarse de forma manual. Sería deseable contar con mecanismos de comunicación asincrónico para satisfacer mejor los requerimientos de los usuarios en estos escenarios.

V. Conclusiones y Trabajos futuros

Como se presentó en este documento, Taverna es una potente herramienta para el diseño, desarrollo y ejecución de workflows científicos. Además de contar con potentes funcionalidades nativas, permite ser extendido con funcionalidades foráneas que incrementan su capacidad y potencia. Algunas de ellas presentadas en este documento.

Por otro lado, Taverna aún no cuenta con algunas características típicas de los workflows y plataformas empresariales, que pueden potenciar el desarrollo de los workflows biológicos, como son la programación orientada a eventos y mecanismos de comunicación asincrónica. Asimismo, existen requerimientos biológicos a los cuales Taverna, hoy día, provee poco o débil soporte, como ser la transformación de formato de datos y la reconfiguración automática de los workflows. En ese sentido, una mayor potencia a las mismas podría tener un impacto significativo y beneficioso.

El próximo paso en este trabajo, es la exploración de las características no soportadas o soportadas débilmente por Taverna y el análisis de su integración, con el fuerte objetivo de ampliar sus funcionalidades y así potenciar la plataforma ^{my}Grid para el desarrollo de experimentos biológicos.

Agradecimientos

Los autores agradecen los comentarios y aportes realizados por la colega Laura González de la Facultad de Ingeniería, Universidad de la República, en la elaboración de este reporte.

REFERENCIAS

[1] D. Hull, K. Wolstencroft, R. Stevens, C. Goble, M. R. Pocock, P. Li, and T. Oinn, "Taverna: a tool for building and running workflows of services," Nucl. Acids Res., vol. 34, no. suppl_2, pp. W729-732, July 2006. [Online]. Available: http://view.ncbi.nlm.nih.gov/pubmed/16

[2] T. Oinn et al., "Taverna: lessons in creating a workflow environment for the life sciences," Concurrency and Computation: Practice and Experience, vol. 18, no. 10, pp. 1067-1100, 2006. [Online]. Available: http://dx.doi.org/10.1002/cpe.993

[3] T. Oinn et al., "Taverna: a tool for the composition and enactment of bioinformatics workflows," Bioinformatics, vol. 20, no. 17, pp. 3045-3054, November 2004. [Online]. Available: http://dx.doi.org/10.1093/bioinformatics/bth361

[4] EMBL Nucleoide Sequence Database, http://www.ebi.ac.uk/embl/ - Julio, 2010

[5] B. Boeckmann et al., "The swiss-prot protein knowledgebase and its supplement trembl in 2003," *Nucl. Acids Res.*, vol. 31, no. 1, pp. 365-370, January 2003. [Online]. Available: http://dx.doi.org/10.1093/nar/gkg095

[6] Align Sequence using ClustalW, http://www.ebi.ac.uk/Tools/clustalw/ - Julio, 2010

[7] BLAST: Basic Local Alignment Search Tool, http://blast.ncbi.nlm.nih.gov/Blast.cgi - Julio, 2010

[8] L. Stein, "Creating a bioinformatics nation." Nature, vol. 417, no. 6885, pp. 119-120, May 2002. [Online]. Available: http://dx.doi.org/10.1038/417119a

[9] MyGrid, http://www.mygrid.org.uk/ - Julio, 2010

[10] Senger M., Rice P., Oinn T., "Soaplab - a unified Sesame door to analysis tools", Proceedings, UK e-Science- All Hands Meeting 2003, p. 509-513, 2003

[11] W. Tan, P. Missier, R. Madduri, and I. Foster, "Building scientific workflow with taverna and bpel: A comparative study in cagrid," Service-Oriented Computing – ICSOC 2008 Workshops, pp. 118-129, 2009. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-01247-1_11

[12] M. D. Wilkinson and M. Links, "Biomoby: an open source biological web services proposal." Briefings in bioinformatics, vol. 3, no. 4, pp. 331-341, December 2002. [Online]. Available: http://dx.doi.org/10.1093/bib/3.4.331

[13] E. Kawas, M. Senger, and M. Wilkinson, "Biomoby extensions to the taverna workflow management and enactment software," *BMC Bioinformatics*, vol. 7, no. 1, pp. 523+, November 2006. [Online]. Available: http://dx.doi.org/10.1186/1471-2105-7-523

[14] T. B. Consortium, "Interoperability with moby 1.0-it's better than sharing your toothbrush!" *Brief Bioinform*, vol. 9, no. 3, pp. 220-231, May 2008. [Online]. Available: http://dx.doi.org/10.1093/bib/bbn003

[15] Jerarquía de objetos Moby, http://moby.ucalgary.ca/RESOURCES/MOBY-S/Objects - Julio, 2010

[16] Jerarquía de servicio Moby, http://moby.ucalgary.ca/RESOURCES/MOBY-S/Services - Julio, 2010

[17] Excel plugin for Taverna 2 now available, http://www.mygrid.org.uk/2009/08/excel-plugin-for-taverna-21-beta-2-now-available/ - Julio, 2010

[18] Taverna – WS-Security support, http://www.taverna.org.uk/documentation/faq/security/ws-security-support/ - Julio, 2010

[19] Taverna - Can I invoke secure services from workflows?, http://www.taverna.org.uk/documentation/faq/security/invoking-secure-services/ - Julio, 2010

[20] D. De Roure, C. Goble, and R. Stevens, "The Design and Realization of the myExperiment Virtual Research Environment for Social Sharing of Workflows", Future Generation Computer Systems, vol. 25, pp. 561-567, 2008

[21] C. Wroe et al., "Recycling workflows and services through discovery and reuse: Research articles," Concurr. Comput. : Pract. Exper., vol. 19, no. 2, pp. 181-194, February 2007. [Online]. Available: http://dx.doi.org/10.1002/cpe.v19:2

[22] P. Fisher et al., "A systematic strategy for large-scale analysis of genotype phenotype correlations: identification of candidate genes involved in african trypanosomiasis." Nucleic acids research, vol. 35, no. 16, pp. 5625-5633, August 2007. [Online]. Available: http://dx.doi.org/10.1093/nar/gkm623

[23] Bechhofer, S., De Roure, D., Gamble, M., Goble, C. and Buchan, I. (2010) Research Objects: Towards Exchange and Reuse of Digital Knowledge. In: The Future of the Web for Collaborative Science (FWCS 2010), April 2010, Raleigh, NC, USA.

[24] E-Labs, http://www.mygrid.org.uk/tools/e-labs/ - Julio, 2010

[25] P. Li et al., "Performing statistical analyses on quantitative data in taverna workflows: an example using r and maxdbrowse to identify differentially-expressed genes from microarray data." BMC bioinformatics, vol. 9, no. 1, pp. 334+, 2008. [Online]. Available: http://dx.doi.org/10.1186/1471-2105-9-334

[26] I. Wassink et al., "Using r in taverna: Rshell v1.2." BMC research notes, vol. 2, no. 1, pp. 138+, July 2009. [Online]. Available: http://dx.doi.org/10.1186/1756-0500-2-138

[27] S. Urbanek, "Rserve - A Fast Way to Provide R Functionality to Applications", Proc. of the 3rd International Workshop on Distributed Statistical Computing (DSC 2003), ISSN 1609-395X, Eds.: Kurt Hornik, Friedrich Leisch & Achim Zeileis

[28] BioRange Software, Rshell, http://janus.cs.utwente.nl:8000/twiki/bin/view/BioRange/BioRangeSoftware - Julio, 2010

[29] R-scripts with the RShell processor, http://www.mygrid.org.uk/usermanual1.7/rshell_processor.html - Julio, 2010

[30] Oasis, 2004. Web Services Security: SOAP Message Security 1.0 (WSSecurity)

[31] RFC 2617: HTTP Authentication: Basic and Digest Access Authentication, http://www.ietf.org/rfc/rfc2617.txt - Julio, 2010

[32] P. Missier, N. W. Paton, and K. Belhajjame, "Fine-grained and efficient lineage querying of collection-based workflow provenance," in EDBT '10: Proceedings of the 13th International Conference on Extending Database Technology. New York, NY, USA: ACM, 2010, pp. 299-310. [Online]. Available: http://dx.doi.org/10.1145/1739041.1739079

[33] Paolo Missier, Satya Sahoo, Jun Zhao, Carole Goble, and Amit Sheth. Janus: from workflows to semantic provenance and linked open data. In Proceedings of The third International Provenance and Annotation Workshop, Troy, NY, U.S.A, 2010.

[34] E-Labs, http://www.mygrid.org.uk/tools/e-labs/ - Julio, 2010

[35] I. Yanai, J. C. Mellor, and C. DeLisi, "Identifying functional links between genes using conserved chromosomal proximity." Trends Genet, vol. 18, no. 4, pp. 176-179, April 2002. [Online]. Available: http://view.ncbi.nlm.nih.gov/pubmed/1193201

[36] M. Wilkinson, B.Vandervalk and E. Luke McCarthy, "SADI Semantic Web Services -- 'cause you can't always GET what you want!", APSCC, pp. 13-18, 2009

[37] EBI_InterProScan2 Workflow, http://www.myexperiment.org/workflows/4.html - Julio, 2010