Entorno de Cómputos Bioinformáticos

Informe de Proyecto de Grado

Christian Schmeichel & Verónica Barber

Tutor: Guzmán Llambías

Cliente: Pablo García

2011

MONTEVIDEO, URUGUAY.

INFORME DE PROYECTO DE GRADO PRESENTADO AL TRIBUNAL EVALUADOR COMO REQUISITO DE EVALUACIÓN DE LA CARRERA INGENIERÍA EN COMPUTACIÓN, UNIVERSIDAD DE LA REPÚBLICA

Resumen del Trabajo

El proyecto realizado tuvo como motivación la construcción de un entorno de automatización de cómputos bioinformáticos que facilitase la labor de los científicos. Con este objetivo en mente, el proyecto abarcó una amplia gama de aspectos que fueron desde la investigación de las herramientas disponibles hoy en día en el mercado para los bioinformáticos y los motores de workflow, el estudio de las problemáticas con las que se enfrentan hoy en día los bioinformáticos hasta la implementación de una herramienta ágil, distribuida y fácilmente extensible.

El proyecto, en primer lugar tuvo una fuerte componente de investigación que permitió entender la temática de la bioinformática, los propósitos que persigue y los desafíos con los que se debe enfrentar. Bajo esta etapa se realizaron cursos del PEDECIBA [Ref36], se leyeron publicaciones y se establecieron los lineamientos que debía seguir el proyecto.

En segundo lugar, con el propósito de diseñar una solución flexible y fácilmente extensible que permitiese modelar los procesos bioinformáticos se estudiaron 5 casos de uso propuestos por el cliente. Dichos casos de uso, aunque simples, permitieron brindar mayor luz a los problemas con los que se enfrentan diariamente los científicos y así, las situaciones a considerar. Estos casos de uso, implementados mediante librerías *Open Source* permitieron incluso ver la falta de documentación con la que se cuenta para poder integrar el conocimiento diseminado en el mundo sobre la temática.

Asimismo, se estudiaron y utilizaron herramientas ampliamente aceptadas en la comunidad bioinformática para el modelado de flujos de trabajo como son *Microsoft Trident* y *Taverna*. Dichas herramientas ofrecen entornos de trabajo flexibles, amigables al usuario y con integración con múltiples sistemas. Sin embargo, estos sistemas carecen de la escalabilidad, propósito de este proyecto.

En tercer lugar, el proyecto se concentró en el estudio de las especificaciones de motores de trabajo que se encuentran actualmente en el mercado. Bajo esta consigna se vieron *WS-BPEL* y *YAWL*. Se consideró utilizar una de las especificaciones pero debido a limitantes en las tecnologías como ser el no contar con una implementación verificada de las mismas en la tecnología Microsoft .NET, se descartaron.

Por último, se implementó y verificó un entorno de trabajo distribuido y escalable que mediante colas de mensaje permitiese la ejecución de procesos bioinformáticos. Esta

herramienta implementó las funcionalidades deseables para dicho entorno las cuales eran ejecución con streaming de datos y la posibilidad de pausar, editar y retomar flujos de trabajo. En forma adicional, se implementó una interfaz gráfica amigable al usuario que permitiese modelar y ejecutar flujos sin requerir conocimientos avanzados de informática.

1. Introducción

La genómica ha visto una explosión masiva en la cantidad de información biológica disponible gracias a los grandes avances de áreas como la biología molecular. Debido a esta explosión masiva surge como una necesidad natural la bioinformática [Ref1].

La bioinformática, la aplicación de la tecnología de las computadoras en la gestión y el análisis de datos biológicos, es un área de investigación interdisciplinaria que actúa como interfaz entre las ciencias biológicas y computacionales. Su meta principal es la de descubrir la riqueza de información biológica que se encuentra escondida para así brindar luz en los fundamentos biológicos de los organismos.

Como resultado, la bioinformática produjo que las computadoras estén siendo utilizadas por los científicos para la recolección, almacenamiento, análisis y combinación de datos.

Si bien en la actualidad existen varios productos informáticos que sirven como apoyo para automatización de tareas, también es común que los científicos desarrollen sus propios scripts para resolver problemas específicos.

La informática no solo ha aportado en cuanto respecta al procesamiento de datos en si y la automatización de tareas. La creación de diversas comunidades para intercambio de información, resultó una herramienta más que beneficiosa para el desarrollo de los distintos trabajos.

1.1 Motivación

La motivación del proyecto surge de la necesidad en el mercado de la implementación de una herramienta flexible que permita automatizar la labor de estos científicos, dado que hoy en día, la mayor parte de las ejecuciones que realizan son mediante scripts *Python* o workflows pesados y poco flexibles.

Tanto los entornos de automatización que existen en la actualidad como los scripts realizados por los propios científicos, presentan diversas limitaciones, entre las cuales se destaca la rigidez para adaptarse a sistemas escalables.

La aparición del procesamiento Cloud o “en la nube” resulta como una oportunidad para abordar los problemas con la escala de datos. Esto implica un cambio estructural y lógico para poder abarcar el cambio de paradigma.
El procesamiento de grandes volúmenes de datos de manera secuencial, ofreciendo resultados intermedios en forma de streaming fue otro de los motivos para realizar este proyecto. En la actualidad, el streaming de datos sobre flujos de trabajos está contemplado de forma muy teórica y resultaría un gran aporte para dicha comunidad.

1.2 Alcance y Resultados Esperados

El alcance del proyecto consiste en el desarrollo de una herramienta de apoyo al desarrollo de experimentos bioinformáticos que permita procesar grandes volúmenes de datos en forma sencilla sirviéndose de procesos de transformación *on-the-fly*. Se pretende desarrollar una herramienta que realice un uso extensivo de *pipes and filters* y que permita diseñar gráficamente la ejecución de cálculos, la extracción de datos, las transformaciones y las diversas salidas permitiendo en cada paso intermedio medianamente pesado, almacenar resultados. Se buscará la ejecución con paralelismo, de forma de poder analizar los cálculos en tiempo real. Para acelerar el procesamiento de los datos se podrán realizar cálculos utilizando GPU [Glo7.1.1], ejecución en Clúster [Glo7.1.2] o Windows Azure [Glo7.1.7].

Los objetivos mínimos del proyecto consisten en el desarrollo de un motor de workflow que permita la ejecución de procesos bioinformáticos y que brinde las funcionalidades de pausa y retoma y streaming. Asimismo, esta herramienta debe tener una arquitectura escalable a entornos cloud y clúster. Por último, la herramienta debe permitir ejecutar servicios web.

1.3 Evolución del Proyecto

Como etapa inicial del proyecto se realizó un relevamiento analizando software de terceros, relevando las funcionalidades y prestaciones ofrecidas.

Dentro de la amplia gama de sistemas de workflow existentes, se analizó el software perteneciente a *Microsoft Research “Trident”* y el software *Open Source* creado por *myGrid Proyect “Taverna“*. Los mismos se describen en la sección 2.3.3 – Herramientas Conocidas.

Para poder abordar el desafío de implementar streaming sobre un motor de workflow, se analizaron trabajos previamente existentes en la comunidad científica. Se tomaron como referencia los trabajos [Ref38] [Ref39] sirviendo éstos como punto de partida o guía práctica para nuestro proyecto. La implementación de streaming se amplía en la sección 4.3.5 – Streaming.

Para poder desarrollar la funcionalidad de Pausar – Editar – Retomar un flujo de ejecución en un motor de workflow, se realizó una implementación propietaria, la cual se describe en la sección 4.3.6 – Pausa y Retoma.

1.4 Resultados Alcanzados

Se implementó un motor de workflow que sigue una arquitectura distribuida y escalable a entornos cloud. La herramienta brinda las funcionalidades de streaming, pausa y retoma, ejecución de servicios web y evaluación de condiciones. Su implementación permite la ejecución de procesos parametrizables con lectura a disco de archivos.

Asimismo, dicha herramienta posee dos interfaces de usuario: una de consola y otra gráfica. La primera interfaz permite al usuario por un lado a través de logs observar el funcionamiento del servidor y por el otro, la ejecución de comandos. La interfaz gráfica permite en un entorno amigable, el modelado y la creación de procesos, la ejecución de los mismos con su correspondiente monitoreo.

1.5 Estructura del Documento

El resto del documento se organiza de la siguiente manera. La sección 2 presenta el marco conceptual. En la sección 3 se describen los requerimientos presentados por el cliente. En la sección 4 se detalla el software implementado. La sección 5 presenta consideraciones y aspectos sobre el trabajo futuro. Por último, en la sección 6 se presentan las conclusiones del trabajo y en las subsiguientes secciones se presentan los adicionales: glosario, referencias y anexos.

2. Marco Conceptual

2.1 Bioinformática

2.1.1 Definición

En las pasadas décadas, los grandes avances en el campo de la biología molecular junto a los realizados en el campo de la genética, han llevado a un aumento explosivo de la información biológica generada por la comunidad científica [Ref1].

Asimismo, la ciencia de la computación ha tenido una evolución sostenida y vertiginosa permitiéndole cooperar con otras ciencias y disciplinas.

Es así, en este contexto, que surge la bioinformática: dándose como la natural respuesta a la necesidad de por un lado mantener bases de datos computarizadas que permitan el almacenamiento, la organización y el indexado de la información biológica generada por una infinidad de fuentes; y por el otro, como la vía para la generación de herramientas especializadas que permitan la correcta visualización y análisis de la información.

Sin embargo, antes de profundizar en el alcance de sus aplicaciones, las problemáticas que enfrenta e incluso, su historia, definamos la bioinformática y las áreas que abarca.

Según el *"National Center for Biotechnology Information"* la bioinformática *es "un campo de la ciencia en el cual confluyen varias disciplinas tales como: biología, computación y tecnología de la información. El fin último de este campo es facilitar el descubrimiento de nuevas ideas biológicas así como crear perspectivas globales a partir de las cuales se puedan discernir principios unificadores en biología"* [Ref2].

Asimismo, La Real Academia Española la define como *"la aplicación de las técnicas informáticas al estudio de la información genética*". Otras acepciones más populares la definen como *"la disciplina que aplica la ciencia de la computación y las estadísticas al campo de la biología molecular"* [Ref3].

Por otro lado, el instituto Pasteur define a la bioinformática como la derivación de conocimiento a través del análisis computacional de datos biológicos, donde esta información biológica puede consistir en el código genético, resultados de experimentos obtenidos de diversas fuentes, estadísticas de pacientes y literatura científica. La

investigación en bioinformática incluye el desarrollo de métodos para el almacenamiento, obtención y análisis de datos. La bioinformática la define como una rama de la biología que se encuentra en rápido desarrollo, altamente interdisciplinaria, que usa conceptos de informática, estadística, matemáticas, química, bioquímica, física y lingüística. Posee amplias aplicaciones en diferentes áreas de la biología y la medicina [Ref4].

Luego, a partir de estas definiciones, nuestra definición de bioinformática es la siguiente:

"la finalidad de la bioinformática es la de aportar las herramientas informáticas que permitan profundizar el conocimiento que se tiene de la biología, bien a través de algoritmias que permitan agilizar los procesos de análisis o a través de sistemas de información que permitan mantenerla accesible para la comunidad científica."

2.1.2 Importancia

Con los niveles de avance que existen hoy en día para la generación de nueva información, la cantidad de datos resultante es totalmente inabarcable para un científico o aun así, para un gran grupo de científicos.

Ya sea un simple proyecto de laboratorio o el conocido "Proyecto Genoma Humano"[Ref37] pueden producir una secuencia de datos que solamente podría ser, ya sea por complejidad o por cantidad, analizable con ayuda de una o un grupo de computadoras.

Un claro desafío que enfrenta la comunidad bioinformática hoy en día es el almacenamiento inteligente y eficaz de esta masa de datos. Es por esto que es necesario proporcionar un acceso fácil y confiable a estos datos.

Si bien el tema del almacenamiento parece inicialmente trivial, los datos recolectados pueden resultar desde pequeños archivos de texto a gigantescas estructuras de archivos con tamaños de varios Terabytes.

Algunas organizaciones tales como NCBI [Ref15] prestan servicios para el procesamiento de datos, permitiendo así una amplia gama de operaciones sobre los mismos. Este servicio resulta particularmente útil para descentralizar y procesar información obtenida en el laboratorio, pero con limitaciones actuales como por ejemplo el ancho de banda

en conexiones vía Internet, el procesamiento de varios Terabytes de información puede resultar también en una tarea compleja.

Según [Ref14], hay tres procesos biológicos centrales en torno al cual se deben desarrollar las herramientas bioinformáticas:

- Secuencia de ADN [Glo7.2.4] que determina la secuencia de proteínas
- Secuencia de una proteína determina la estructura de proteínas
- Estructura de una proteína determina la función de proteínas

2.2 Aplicaciones

La bioinformática posee numerosas aplicaciones al campo de la biología dado que no sólo brinda soporte para el procesamiento de datos y el modelado y la simulación de sistemas biológicos, sino que también sirve de fuente para la generación de nueva información a través del cruzamiento de datos y la producción de biochips [Glo7.2.2].
Sus aplicaciones se pueden separar en tres ramas bien definidas:

1. Brindar la infraestructura y los sistemas necesarios para procesar la información: redes, bases de datos o imágenes.

2. Permitir el modelado y simulación de asuntos biológicos del campo de la biología molecular, tales como redes de neuronas artificiales.

3. Dar soporte a través de materiales y modelos biológicos utilizados como base de sistemas computacionales, entre ellos biochips.

2.3 Desafíos

La bioinformática se centra predominantemente en tres grandes áreas de información de la biología molecular: estructuras macromoleculares [Glo7.2.9], secuencias genómicas [Glo7.2.10] y los resultados de experimentos genómicos como ser la expresión de genes [Glo7.2.11]. En forma agregada, la bioinformática también estudia las vías metabólicas [Glo7.2.12], los árboles taxonómicos [Glo7.2.13] y la interacción entre proteínas.

El desafío de la bioinformática se encuentra en buscar mediante el empleo de una amplia gama de técnicas computacionales, brindar luz sobre temas tan diversos como la geometría macromolecular, la reconstrucción de árboles filogenéticos [Glo7.2.14] y la

predicción de estructuras y funciones proteicas entre otros. La disciplina ha permitido numerosos avances en el conocimiento científico de los organismos y ha brindado numerosos beneficios prácticos como ser el diseño de medicamentos, la predicción de enfermedades y la reconstrucción de genéticas poblacionales.

Debido a la dificultad que presenta el análisis de datos, la alineación de secuencias o la predicción de estructuras de proteínas (entre otros), se utilizan algoritmos informáticos para realizar estas tareas.

Si bien se pretende obtener órdenes de ejecución polinomiales para estos algoritmos, en algunas ocasiones éstos pueden resultar exponenciales. Analizaremos a continuación algunos aspectos relevantes.

COMPLEJIDAD ALGORÍTMICA

Elegir un algoritmo correcto para el procesamiento de datos es una tarea fundamental. Por ejemplo, un algoritmo de tipo "*Brute Force*" puede resolver un problema dado con exactitud, pero al tener millones de datos a procesar se tardaría años o incluso décadas en obtener un resultado.

Por este motivo es que se utilizan varios enfoques diferentes, entre los cuales se presentan:

- *Brute Force*: Algoritmo simple pero lento, pues prueba todas las combinaciones disponibles.

- *Branch & Bound*: Se realizan podas de alternativas no viables.

- *Divide & Conquer*: Divide cada problema en subproblemas más simples que se resolverán de forma independiente.

- *Machine Learning*: Utilización de datos estadísticos previamente obtenidos para mejorar la búsqueda.

- *Algoritmos Randómicos*: Se utiliza el azar para la toma de decisiones.

- *Programación Dinámica*: Subdivisión en problemas más pequeños realizando los cálculos en base a resultados anteriores.

En la bioinformática también existen muchos problemas NP-Complete para los cuales se tienen tiempos de ejecución exponenciales.

Dos ejemplos de problemas NP-Complete son por ejemplo:

- Double Digest Problem [Ref 34], donde a partir de tamaños de fragmentos dados y puntos de cortes de dos enzimas de restricción en secuencias de ADN, se debe reconstruir las secuencias de ADN originales.

- Primer Selection Problem [Ref 35], que busca minimizar el número de primers [Glo7.2.15] para realizar un PCR [Glo7.2.16].

2.4 Bases de Datos Biológicas

Se define una base de datos biológica, como una biblioteca de información sobre datos biológicos recogida de diversas fuentes como experimentos científicos, literatura publicada o análisis computacional. En resumen, contienen información de un amplio espectro de áreas de la biología.

Los datos almacenados deben poder ser accedidos de una manera simple, clara y consistente. También es habitual que el contenido de varias bases de datos tenga que ser accedido en forma simultánea y de manera correlacionada. Con este fin se han desarrollado algunos lenguas especiales tales como “Retrieval System (SRS)” [Ref17] y el sistema “Entrez” [Ref16].

Existe una clasificación para estas bases de datos, las cuales se pueden clasificar entre “primarias” y “secundarias”.

Las bases de datos primarias contienen secuencias de ADN y de proteínas, estructuras de proteínas y perfiles de expresión de genes y proteínas. Cada registro de estas bases de datos contiene una secuencia y su correspondiente "anotación", realizada generalmente por el propio científico que las obtiene. Se dice que estas bases de datos están en un formato crudo, pues no constan de análisis sobre los datos obtenidos.

Las bases de datos secundarias o procesadas se llaman así porque contienen los resultados del análisis de los recursos primarios, con inclusión de información sobre

patrones de secuencias o motifs [Glo7.2.17], variantes, mutaciones y relaciones evolutivas.

2.5 Aplicaciones Biológicas

Las aplicaciones biológicas surgieron como forma de ayuda para el procesamiento de la cantidad de datos almacenados. Una vez que la información obtenida se almacenara de forma coherente y de fácil acceso para la comunidad científica, el siguiente paso a seguir es el de proporcionar métodos eficientes para la extracción y procesamiento de la información significativa de la masa de datos.

Debido a que en la mayoría de los casos, los biólogos no se tratan de personas especializadas o cuentan con conocimientos avanzados de informática, es necesario que las herramientas biológicas cuenten con un funcionamiento intuitivo y ameno para los usuarios. Otro aspecto a ser tomado en cuenta, es que dichas herramientas sean de fácil acceso, pudiendo así, ser utilizadas por toda la comunidad científica.

Algunos institutos, como el EBI (European Bioinformatics Institute) o el NCBI (National Center for Biotechnology Information, EE.UU.) proporcionan herramientas online y de forma gratuita para cualquier usuario.

Dentro de estas herramientas se podría hacer una clasificación por:

- Herramientas para búsqueda de similitud:

Este conjunto de herramientas pueden ser utilizadas para identificar las similitudes entre secuencias de estructura y función desconocidas. Las secuencias homólogas son secuencias que están relacionadas por la divergencia de un antepasado común. De esta manera, se puede medir el grado de similitud entre dos secuencias, mientras que la homología es una relación que puede ser solamente verdadera o falsa.

- Análisis de función de proteínas:

Este conjunto de programas permiten comparar secuencias de proteínas pertenecientes a bases de datos secundarias que contengan información sobre motifs, firmas y dominios de proteínas. El hecho de obtener un alto porcentaje de hits ante los patrones presentados en alguna de las bases de datos, permitiría aproximar la función bioquímica de dichas proteínas.

- Análisis estructural

El análisis estructural es un análisis basado en la comparación de la forma de la proteína, tratando de establecer equivalencias entre dos o más estructuras. La función de una proteína depende de su secuencia y de estructura. Es por este motivo que la determinación de la estructura 2D/3D una proteína es crucial en el estudio de su función. Las herramientas de análisis estructural se basan en la comparación de estructura para poder deducir funcionalidades.

2.6 Workflows científicos

2.6.1 Definición

En los últimos años, las necesidades de la comunidad científica han crecido tanto en potencia de cálculo como en complejidad introducida en los procesos. De esta manera, trabajar con pequeños procesos sincronizados de forma manual se transformó en una tarea de gran complejidad.

Inicialmente se comenzó sincronizando y automatizando la ejecución de pequeños scripts programados en diversos lenguajes, aliviando así la carga de trabajo introducida por tareas repetitivas, pero distando ésta de ser una solución eficiente.

En la actualidad, varios workflows científicos se centran en técnicas de computación Grid. Ésta es una forma de computación distribuida y escalable, generalmente con recursos heterogéneos y de control descentralizado. De esta manera se puede utilizar la potencia de varios sistemas informáticos para apoyar una tarea específica, reduciendo así tiempos y costos asociados.

Las aparición de las arquitecturas orientadas a servicios web significaron un avance fundamental para los workflows científicos, pues permitieron realizar invocaciones de servicios de terceros vía internet.

Ante esta necesidad se han creado una variedad de sistemas de workflow científicos, los cuales ayudaron en esta tarea, pero difiriendo en funcionamiento respecto a los workflows empresariales utilizados hasta el momento.

Para poder entender mejor a que se refiere cuando se habla de workflows científicos, se presentan algunas definiciones.

Por ejemplo, en [Ref 31] se menciona:

"*Utilizamos el término workflow científico como un término general para describir una serie de actividades estructuradas y cálculos que surgen en la resolución de problemas científicos*".

Por otro lado, en [Ref 32] se define como:

"*Un workflow científico es el proceso automatizado que combina datos y procesos en un conjunto estructurado de pasos, para así poner en práctica soluciones computacionales para un problema científico.*"

Esta última definición será la que nosotros tomaremos como válida.

Una vez vistas estas definiciones, uno se vería tentado en suponer que los workflows científicos y empresariales podrían tratarse de la misma cosa, pero abarcando temáticas de trabajo diferentes. Es por esto que en la siguiente sección nos concentraremos en destacar algunas diferencias que existen entre ambos tipos de workflows.

2.6.2 Workflows científicos versus Workflows empresariales

Tal como se mencionó en la sección anterior, la forma de trabajar y flujos de ejecución de tareas en el ámbito empresarial y científico difieren en ciertos aspectos. Es por esto que mayoritariamente no es posible utilizar worflows empresariales en el ambiente científico.

En esta sección, basados en el trabajo de [Ref5] describiremos brevemente en que radican dichas diferencia y cuáles son las realidades en cada caso.

En la tabla [tabla 1] se detallan algunas diferencias entre ambos tipos de workflows, los cuales se discutirán en las siguientes secciones.

	Workflows Científicos	Workflows Empresariales
Documentación	Escasa, estandarización de patrones de diseño casi inexistente.	Amplia documentación, estandarización de patrones de diseño.
Modelo Conceptual y Diseño	Workfow orientado a datos. Concepto de flujo de datos.	Workflow orientado a control. Orden entre tareas.
Objetivos	Optimización de resultados y datos intermedios.	Optimización de procesos.

Tabla 1 - Workflows científicos vs. Workflws empresariales

ESTADO DEL ARTE

Actualmente la literatura referente a workflows científicos no se enfoca en proporcionar patrones característicos que ayuden a entender la especificación o implementación de estos. Es por este motivo que los workflows científicos resultan en varias ocasiones más difíciles de abordar o comprender su estructura y funcionamiento.

Como se describirá en secciones posteriores, el modelado y flujo de trabajo de uno y otro enfoque se realizan de formas diferentes. Es por este motivo que no en todos los casos es posible utilizar los ya conocidos patrones de diseño para workflows empresariales sobre workflows científicos.

Para workflows empresariales, existen iniciativas reconocidas y bien definidas [Ref10] las cuales sirven como base para el modelado de flujos. Para los workflows científicos en cambio, existen diversas propuestas para abordar los temas aún no resueltos de una forma estandarizada.

MODELO CONCEPTUAL Y DISEÑO

Desde el punto de vista científico, la problemática existente con los workflows empresariales, es que los mismos centran su foco en cómo y en qué orden se han de realizar las actividades, mientras que en los workflows científicos existe una orientación mayor a los resultados de cada una de ellas, sin importar tanto el orden de ejecución.

En general, los patrones de workflow son muy conocidos y ampliamente aceptados en los procesos de negocios. Ahora, cuando los flujos de trabajos se mueven al mundo científico, teniendo en cuenta soporte a gran escala, aumento de complejidad, tolerancia a fallas (entre otros), el enfoque exigido o la forma de concebir estos procesos debe ser diferente.

El cumplimiento de un proceso de negocio se trata de un conjunto de reglas de negocio claras dentro de un contexto de jerarquía de la organización, un conjunto de normas y reglas claras de coordinación entre las personas involucradas en el proceso en sí.

En los procesos científicos en cambio, generalmente son pocas las personas involucradas en el mismo proceso. La mayoría de los procesos científicos implican sólo una persona, pero sí varias unidades informáticas diferentes y datos complejos.

Otra necesidad típica es la de correr el mismo proceso varias veces con un conjunto diferente de datos y distintas unidades informáticas. El o los científicos involucrados tienen finalmente la tarea de examinar la gran cantidad de datos resultantes. Es por esta razón que resulta justo decir que una de las ideas clave al mencionar workflows científicos es que "los workflows científicos son orientados a datos".

Tal y como se menciona en [Ref5] modelar workflows orientados a control usando solamente orientación a datos puede "converger rápidamente en workflows complejos, difíciles de entender, reusar, reconfigurar o mantener, lo cual dificultaría aún más la comprensión de los procesos científicos por los propios científicos".

Es sabido que los workflows empresariales proveen primitivas específicas del modelado orientado a control, como lo son la representación de secuencias, ramificaciones o elecciones de caminos. Sin embargo, este tipo de workflows no facilitan las tareas repetitivas o concurrentes, las cuales resultan fundamentales para el desarrollo de los workflows científicos. Como se puede apreciar en la Imagen1, la interpretación de un flujo dado puede variar según el ambiente dado.

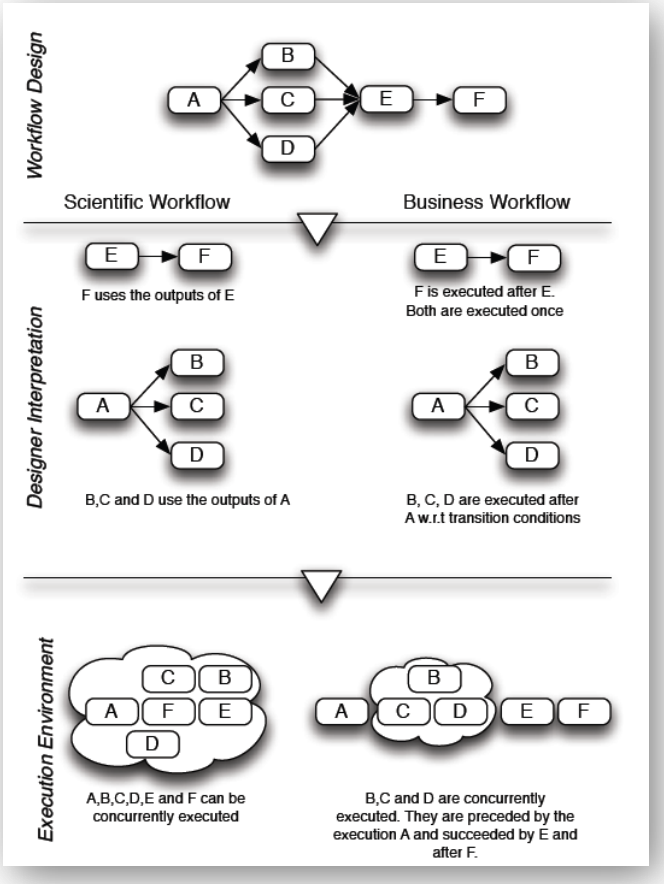

Imagen 1 - Comparación entre Workflows Científicos y Empresariales [Ref5]

Analizando los flujos de trabajo, resalta la idea del orden. Como diferencia fundamental entre ambas iniciativas, los workflows científicos se basan en un modelado de los procesos del flujo de datos, mientras que los flujos de trabajo empresariales por lo general tratan de expresar un orden parcial de tareas.

Tal como se ve en la Imagen1, hay una dependencia de datos entre la entrada y salida de dos tareas consecutivas de un flujo de trabajo científico, mientras que en el flujo de trabajo empresarial, esta dependencia representa una relación de orden parcial entre tareas. Otro aspecto a destacar, es que en el modelado orientado a datos, no existe una imposición de orden entre tareas. Una cantidad dada de tareas pueden ser procesadas concurrentemente en los workflows científicos, mientras que en los workflows empresariales, solo pueden ser ejecutadas concurrentemente las tareas que pertenecen a caminos de ejecución diferentes.

ENFOQUE

Dejando por fuera la definición formal del proceso y su ejecución, los sistemas de workflow científico son los que brindar respuestas integrales a preguntas tales como:

- ¿Cuáles son los insumos utilizados para crear el producto final?
- Teniendo dos productos, ¿fueron estos derivados de los mismos datos originales?

Intuitivamente hablando, los científicos se preocupan mucho acerca de los pasos intermedios, resultados y datos de un proceso científico. En los workflows empresariales, los empresarios están más interesados en saber qué partes de sus procesos se pueden optimizar basados en ciclos anteriores para reducir los costes de mantenimiento. Por lo tanto, la gestión de datos y el proceso que los involucra, siendo estas características típicas de ambos flujos de trabajo, se construyen sobre la base de expectativas de usuarios diferentes.

2.6.3 Herramientas conocidas

Como se mencionó en la sección 1.3 - Evolución del Proyecto, se realizó un relevamiento de las herramientas existentes en el mercado, relevando en cada caso, funcionalidades y prestaciones ofrecidas.

Dentro de la amplia gama de sistemas de workflow existentes, analizamos el software perteneciente a Microsoft Research “Trident” y el software open source creado por myGrid Proyect “Taverna “.

La razón por la cual se seleccionaron estos dos sistemas como base del análisis fue, en caso de “Trident” por tratarse de un software que utiliza una plataforma Microsoft, punto que resultó de vital importancia para establecer un marco de referencia con el software a construir.

En el caso de “Taverna”, el mismo se analizó por tratarse de un software aceptado popularmente por la comunidad científica.

En la siguiente tabla [Tabla 2- Taverna vs. Trident] se destacan algunas características de ambos sistemas de workflow.

	Taverna	Trident
Lenguaje de programación	JAVA	C#
Plataforma	Multiplataforma	Microsoft
Open Source	Si	Si
Interfaz Gráfica	Si	Si
Modos de ejecución	Desktop, Línea de comandos, Como servidor remoto	Desktop, Como servidor remoto
Extensible	Si	Si
Publicación de trabajos y resultados	Integración con MyExperiment y formatos Office	Integración con MyExperiment y formatos Office
Documentación y soporte	Documentación y soporte ofrecidos en página web	Documentación y soporte por Microsoft Research

Tabla 2- Taverna vs. Trident

2.6.4 Taverna

Taverna se trata de un sistema de workflow Open Source escrito en JAVA, creado por myGrid Proyect y financiado por la OMII-UK. [Ref8]

Taverna es un lenguaje de workflow orientado a datos, diseñado para apoyar la automatización de procesos complejos, basados en servicios y procesamiento de gran volumen de datos. A su vez, ha sido aplicado en campos diversos como la bioinformática, la astronomía, la investigación médica o música.

El mismo tiene como base el sistema Taverna Engine, utilizado por Taverna Workbench y Taverna Server, herramientas para interfaz de usuario como la que se muestra en la Imagen2 y la posibilidad de ejecución remota de workflows desde el servidor.

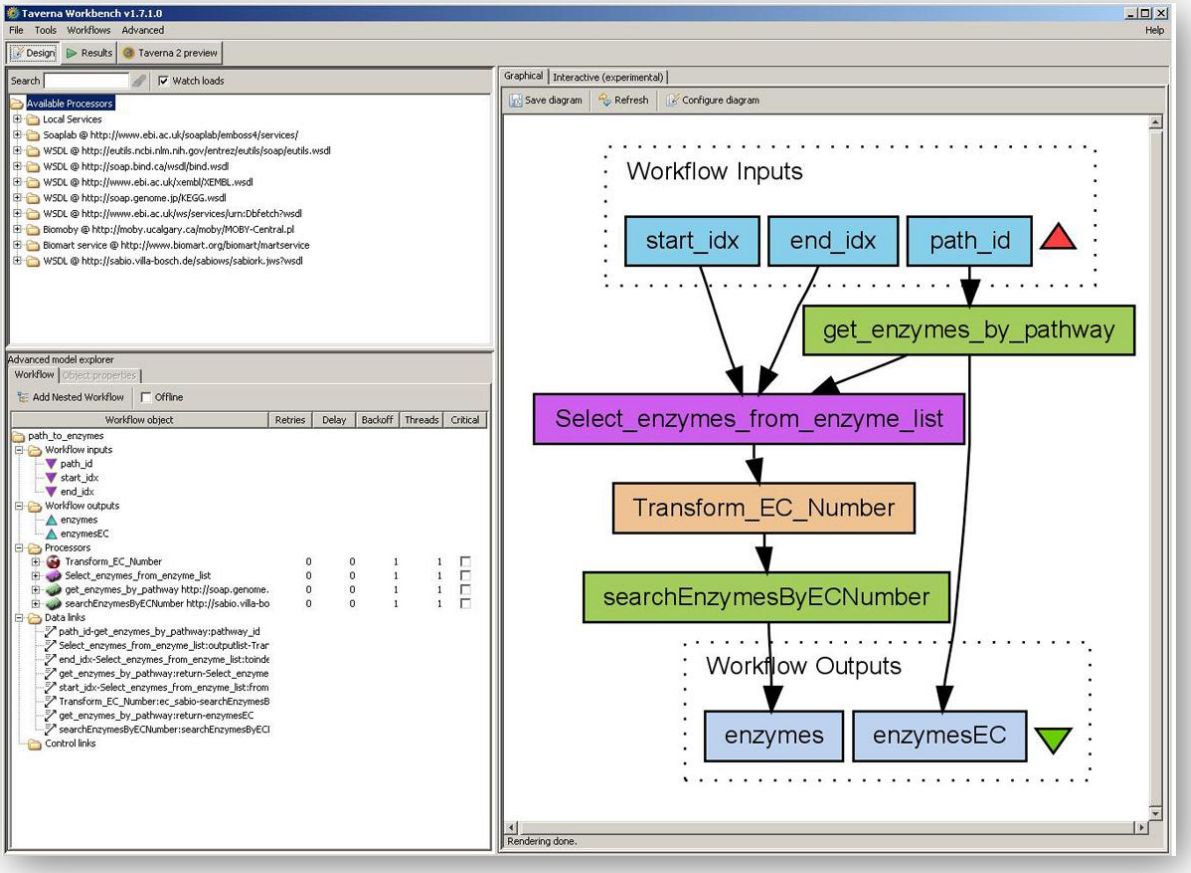

Imagen 2 - Taverna Interfaz Gráfica [Ref8]

Tal como se ve en la Imagen3, si bien la ejecución se realiza de forma descentralizada, un patrón de fachada se utiliza para proporcionar los componentes externos, tales como los componentes de presentación de resultados.

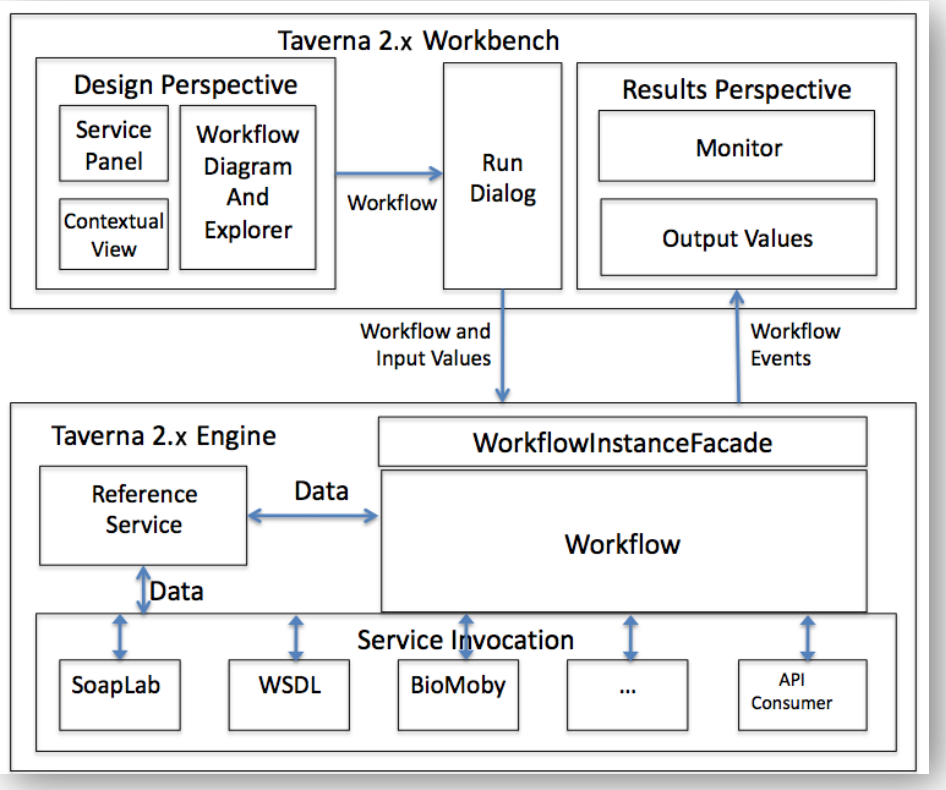

Imagen 3 - Arquitectura Taverna [Ref8]

El sistema de workflow Taverna es extensible, posibilitando adicionar funcionalidades según las necesidades del usuario. La integración con la comunidad My Experiment permite compartir tanto datos como proyectos de forma fácil y fluida, debido a que los creadores de Taverna son también los creadores de dicha comunidad.

Cuenta con una amplia documentación y soporte, principalmente ofrecidos a través de su portal Web.

El hecho de que este software cuente con una interfaz de usuario, arquitectura distribuible, invocación de servicios de terceros y distintas formas de invocación, resulta en un buen punto de análisis para los aspectos de este proyecto.

2.6.5 Trident

El sistema de workflow Trident forma parte del "Project Trident" perteneciente a Microsoft Research. Este sistema se basa en Windows Workflow Foundation (WF), una plataforma flexible y potente para implementar flujos de trabajo.[Ref7]

Trident incluye dos aplicaciones, Trident Composer y Trident Management Studio. La combinación de ambas Imagen4 se conoce como Trident Workbench, sistema que proporciona herramientas para crear, ejecutar y administrar los flujos de trabajo.

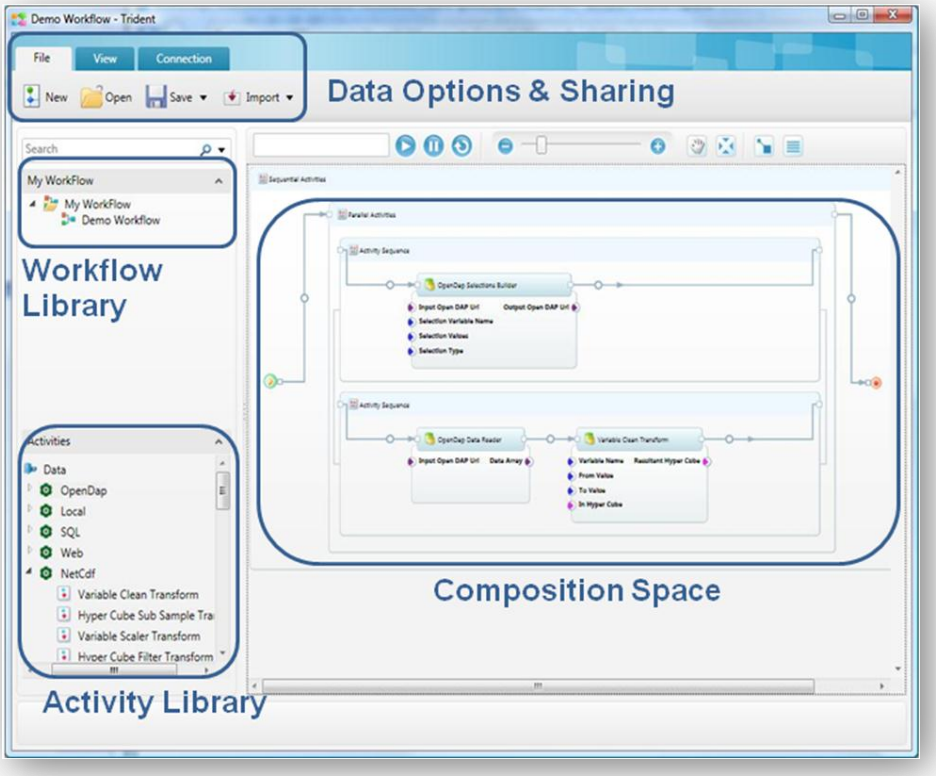

Imagen 4 - Workflow Composer [Ref7]

Asimismo, este sistema incluye la posibilidad de extender el conjunto estándar de funcionalidades mediante la implementación de funcionalidades personalizadas. En la Imagen5 se destacan las diferentes áreas funcionales.

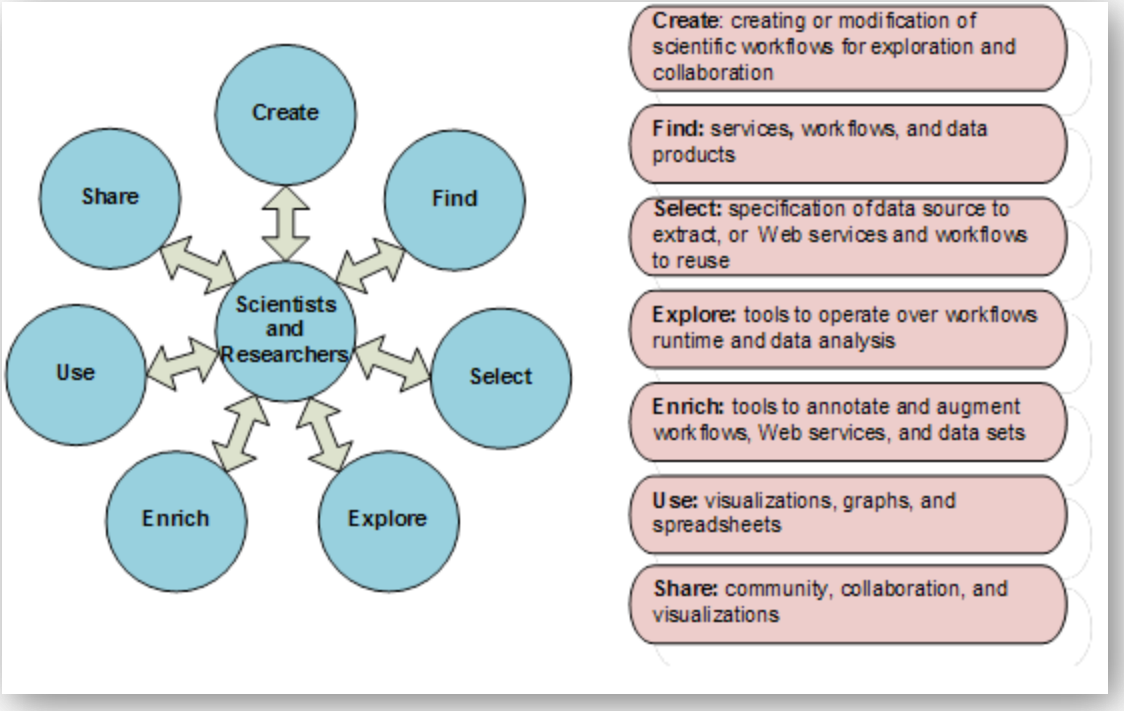

Imagen 5 - Áreas Funcionales WorkFlow [Ref7]

- **Crear**: Creación o modificación de workflows científicos para exploración y colaboración.
- **Encontrar**: Encontrar Servicios, workflows y productos.
- **Seleccionar**: Especificar las fuentes de datos, Web Services o workflows a reutilizar.
- **Explorar**: Herramientas para trabajar con workflows y análisis de datos.
- **Enriquecer**: Herramientas para comentar y mejorar workflows, Web Services y sets de datos.
- **Utilizar**: Visualización, gráficas y hojas de cálculo.
- **Compartir**: Comunidad, colaboración y visualización.

Trident presenta una arquitectura lógica separada en capas Imagen6 dentro de las cuales se encuentran:

- **Visualización**: Esta capa no solo incluye la interfaz de usuario, sino que también soporta la visualización de datos a través de todas las fases de una aplicación workflow, incluyendo el diseño, ejecución, distribución y archivado.

- **Servicios de Runtime**: Los servicios se comunican entre sí por una arquitectura tipo Blackboard productor-suscriptor. De esta manera los desarrolladores pueden crear servicios personalizados apoyados en Windows Workflow y esta arquitectura.

- **Manejo de datos**: La capa de gestión de datos permite la abstracción de la tecnología de almacenamiento de datos subyacente. El borde superior se comunica con la capa de Servicios de ejecución y maneja la mecánica de pasar los datos hacia y desde el almacenamiento de datos seleccionado.

- **Data Storage**: Los datos pueden ser almacenados en una variedad de formas, incluyendo: Microsoft SQL Server database o Bases de datos cloud (SQL Server Data Services (SSDS) o Amazon Simple Storage Service (S3)).

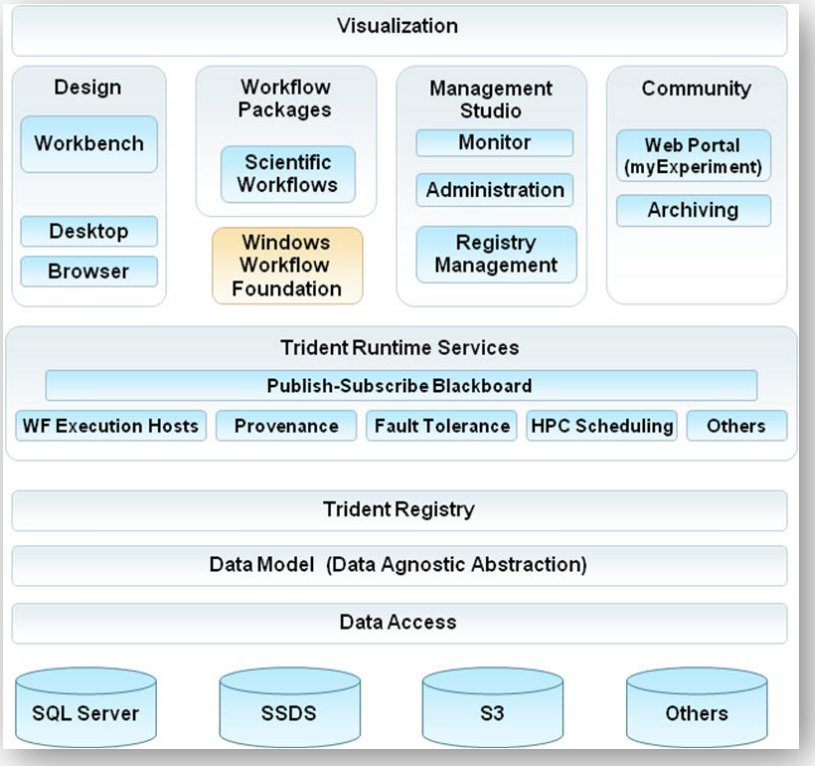

Imagen 6 - Arquitectura lógica Trident [Ref7]

Por otro lado, al igual que otros sistemas de workflow, también permite exportar flujos de trabajo en formato Open Packaging Convention (OPC) y subirlos a la comunidad My Experiment, sitio en el cual se comparten proyectos y resultados de la comunidad científica en forma fácil y fluida. Trident también presenta la posibilidad de embeber dichos trabajos dentro de archivos de Microsoft Office. De esta manera se facilita la forma de compartir información dentro de la comunidad científica.

2.6.6 Microsoft Biology Foundation

La plataforma .NET Bio de Microsoft, antes denominada Microsoft Biology Foundation es una herramienta bioinformática que funciona como una extensión de Framework Microsoft .NET que estuvo originalmente destinada a la investigación genómica.[Ref19]

Actualmente dicha herramienta ofrece las siguientes funcionalidades:

1. Una serie de parsers para formatos de archivo básicos del área bioinformática. Estos formatos comprenden fasta, multifasta, entre otros.

2. Un conjunto de algoritmos amplio que permite manipular secuencias de ADN, ARN y proteicas.

3. Un conjunto de conectores que permiten la integración con servicios web provistos por NCBI. Un ejemplo de estos conectores es la integración con el servicio web de ejecución de algoritmos BLAST.

Como parte de la introducción al área de la bioinformática, se realizó una investigación sobre 5 herramientas bioinformáticas de uso extendido en la comunidad y de una serie de laboratorios que ofrece Microsoft Biology Foundation.

Se investigaron 5 librerías ofrecidas por la comunidad OpenSource [Ref20] para la ejecución de procesos bioinformáticos. Estos procesos o casos de uso simples fueron cruciales dado que permitieron entender el tipo de información a manipular, los procesos que se necesitan modelar y la flexibilidad con la que debe contar para poder especificar el conjunto de variables que maneja un proceso típico bioinformático.

Para cada una de las librerías, se realizó una investigación de su funcionamiento, se realizaron pruebas y se intentó llegar a un diseño de XML [Glo7.1.8] que permitiese al unirlos, el modelado en conjunto de los 5 casos de uso.

Estos 5 casos de uso se describen en la sección 9.3 – Trabajos Relacionados.

2.7 Workflows empresariales

Al tratarse éste de un proyecto asociado a un flujo de trabajo, se decidió analizar las especificaciones de lenguajes de workflows previamente existentes. De esta manera se podría decidir si utilizar uno de estos lenguajes o realizar una implementación propietaria.

Debido a su popularidad y uso en el mercado, se decidió analizar las especificaciones de los lenguajes BPEL y YAWL.

Se presentará a continuación una breve reseña de los lenguajes analizados junto a sus características.

2.7.1 YAWL

Este lenguaje YAWL (Yet Another Workflow Language), fue desarrollado por Wil van der Aalst (Eindhoven University of Technology, the Netherlands) y Arthur ter Hofstede (Queensland University of Technology, Australia) en el año 2002. Se trata de un ambiente Open Source desarrollado en colaboración con la industria.[Ref9]

YAWL es un lenguaje de orquestación y se basa por un lado en Redes de Petri, el cual es un sistema de eventos discretos que permite concurrencia y presenta una representación gráfica. Por otro lado se concentra en la utilización de patrones de workflows definidos. [Ref10] Tal como definen sus autores [Ref11]

"Las redes de Petri puede capturar un buen número de patrones de workflow de control definidos, pero carece de apoyo para patrones de múltiples instancias, patrones de cancelación o el generalizado or-join. YAWL extiende entonces las redes de Petri con construcciones dedicadas para hacer frente a estos patrones".

YAWL ofrece las siguientes características distintivas:

- **Soporte de patrones:** YAWL ofrece soporte exhaustivo para los patrones de flujo de control. Es el lenguaje más poderoso para especificación de procesos capturando las dependencias de control de flujo.

- **Esquemas XML:** La perspectiva de los datos en YAWL se captura mediante el uso de esquemas XML, XPath y XQuery.

- **Sin ambigüedad**: YAWL tiene un fundamento formal propio. Esto hace que sus especificaciones sean sin ambigüedades y que la verificación sea automatizada.

- **Manejo de excepciones:** YAWL ofrece soporte único para manejo de excepciones, tanto para las existentes como las que no se previeron en tiempo de diseño.

- **Workflows dinámicos:** YAWL ofrece un soporte único para workflows dinámicos a través del enfoque de Worklets. Los flujos de trabajo pueden así evolucionar en el tiempo para satisfacer las nuevas necesidades.

- **Fácil Implementación:** YAWL aspira a una implementación fácil. Ofrece una serie de instaladores automatizados y un entorno de diseño gráfico intuitivo.

- **Modelado de procesos:** A través del componente BPMN2YAWL, los modelos BPMN se puede asignar al ambiente YAWL para su ejecución.

- **Arquitectura orientada a servicios:** La arquitectura de YAWL es orientada a servicios y por lo tanto se puede remplazar los componentes existentes componentes propios o ampliar el entorno con nuevos componentes desarrollados.

- **Asignación a distintos actores:** Las tareas en YAWL se pueden asignar a participantes humanos, servicios web, aplicaciones externas o a clases Java.

Para la utilización de YAWL se ha creado un editor con interfaz gráfica, cuyo funcionamiento se describe brevemente en la Imagen7. Este permite al usuario crear flujos de trabajo sin tener que crear especificaciones en lenguaje XML.

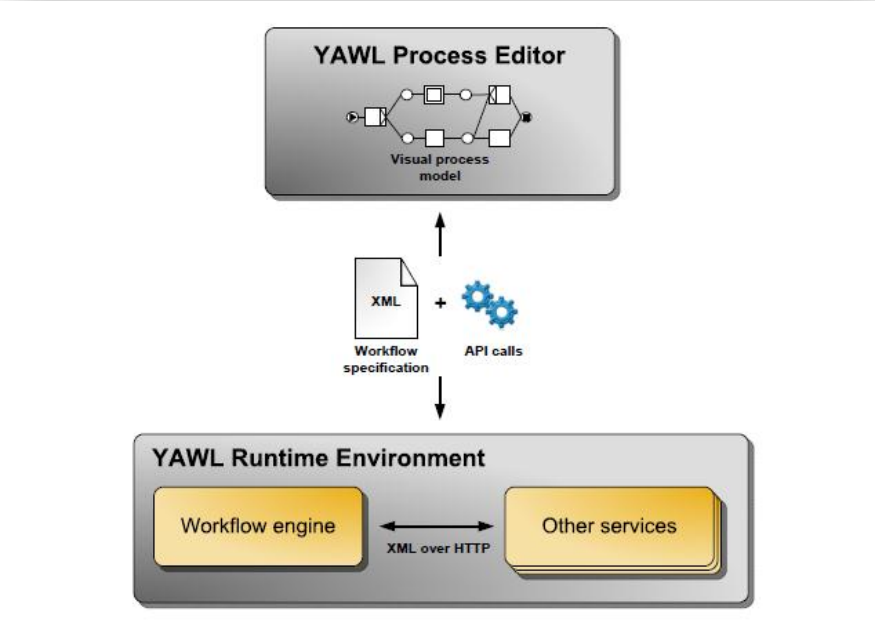

Imagen 7 - Componentes YAWL [Ref11]

Tomando en cuenta la ejecución de un flujo, cada elemento de trabajo tiene un ciclo de vida asociado que puede atravesar las diferentes etapas que se detallan en la Imagen8. Esta figura resulta una de las claves fundamentales para el entendimiento de la ejecución de una tarea y el funcionamiento del motor.

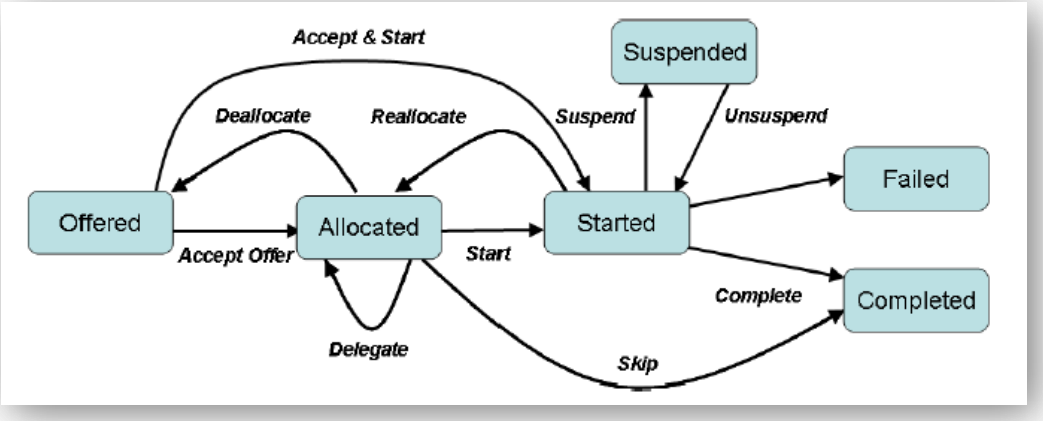

Imagen 8 - Ciclo de vida de una elemento de trabajo [Ref11]

2.7.2 WS-BPEL

WS-BPEL (Web Services Business Process Execution Language) es un lenguaje para describir formalmente procesos de negocio y protocolos de interacción de negocios.

WS-BPEL 2.0 fue aprobado como un estándar OASIS en abril de 2007. Define un modelo y una gramática para describir el comportamiento de un proceso de negocio basado en la interacción entre un proceso y otros procesos involucrados. La interacción con cada proceso asociado se produce a través de interfaces de Web Services. El proceso de WS-BPEL define cómo las interacciones de múltiples servicios se coordinan para lograr un objetivo de negocio, así como los estados y la lógica necesaria para la coordinación.

El mismo es un lenguaje de orquestación y fue diseñado para extender el modelo de la interacción de servicios web para apoyar las transacciones comerciales.

Algunos de los fundamentos para utilizar BPEL se presentan como [Ref12] y [Ref13]:

- **Proporciona un lenguaje estándar para la industria permitiendo la expresión de los procesos de negocio:** Al igual que con su semántica rica y completa, WS-BPEL representa un estándar que ha experimentado un desarrollo riguroso para la industria hacia la solución de las necesidades complejas, lo que resulta en una solución de orquestación completa.

- **Aprovechamiento de un conjunto de habilidades y lenguaje común:** En lugar de utilizar complejas tecnologías patentadas, WS-BPEL implementa buenas prácticas, patrones, experiencia y formación para ser aprovechados por una variedad de proveedores, así como acceso a varios recursos para utilizar WS-BPEL y su tecnología.

- **Lógica de negocios abstracta:** La lógica de negocios puede ser diseñada para ser independiente del proceso y reutilizable. El proceso de negocio asume la responsabilidad y coordinación del estado, liberando a los servicios de una serie de restricciones de diseño. Además, la lógica de los procesos de negocio está centralizada en un solo lugar, en lugar de ser distribuidos a través de los múltiples servicios.

- **Diseñado para adaptarse de forma natural a la pila de Web Services:** En WS-BPEL los procesos de negocio interactúan con los servicios a través de invocaciones de Web Services y ellos mismos son exteriorizados como Web Services. Esta composición recursiva permite a un proceso BPEL aprovechar la interoperabilidad proporcionada por los niveles inferiores de la pila de Web Services, tales como WSDL, SOAP y WS-Addressing.

- **Expresado completamente en XML:** Como los procesos de negocio de WS-BPEL se expresan en XML, son fácilmente legibles por una persona y pueden ser interpretados por cualquier herramienta de procesamiento de XML. Esto también permite que sean producidos y consumidos dentro de la pila de XML.

- **Utilización y se extensión de WSDL 1.1:** WS-BPEL utiliza y extiende WSDL para proporcionar y consumir Web Services de una manera abstracta, utilizando WSDL para definir interfaces de servicios.

- **Interoperabilidad entre los procesos que interactúan:** Muchas implementaciones tendrán múltiples plataformas de orquestación, debido a que estas cuentan con herramientas y aplicaciones embebidas. WS-BPEL proporciona un estándar común que proporciona la interoperabilidad entre las diferentes plataformas y los procesos que se ejecutan en ellos.

Así como BPEL ha sido adoptado por varias empresas del mercado, de la misma manera han surgido distintas implementaciones de motores BPEL en diferentes sistemas operativos y lenguajes de programación. Centrando el foco en tecnologías Microsoft, se encuentra BizTalk Server, el cual se tomó como una opción posible.

2.8 Cloud Computing

La técnica de Cloud Computing presenta un modelo que permite proveer a demanda mediante la red el acceso a un pool de recursos computacionales configurable. Este pool de recursos que puede consistir en acceso a redes, servidores, fuentes de almacenamiento, aplicaciones y servicios, se encuentra disponible al usuario mediante una mínima gestión y requiriendo una mínima interacción con el proveedor [Ref 33].

2.9 Hadoop

Hadoop, un framework que permite el procesamiento distribuido de grandes bancos de datos sobre clústers de computadoras utilizando un modelo de programación simple, está diseñado para permitir que el software que lo utiliza, escale de la ejecución en una computadora, a la ejecución en miles de computadoras, donde cada una ofrece almacenamiento y procesamiento en forma local. Inclusive, el framework permite la detección y manejo de errores a nivel de capa de aplicación, ofreciendo así en forma adicional la funcionalidad de alta disponibilidad. [Ref 24]

Hadoop Distributed File System (HDFS)

HDFS es el sistema de almacenamiento más utilizado por las aplicaciones Hadoop. Un cluster HDFS consiste de un "NameNode" que maneja toda la metadata del sistema y varios "DataNodes" que mantienen la información.

Map Reduce

Hadoop Map Reduce se trata de un framework para escribir fácilmente aplicaciones que procesan grandes cantidades de datos (varios terabytes de tamaño) en paralelo, sobre grandes clusters (miles de nodos) sobre un sistema fiable y tolerante a fallas.

Hadoop Streaming

Hadoop Streaming es una utilidad integrada a la distribución de Hadoop. Esta utilidad permite crear y ejecutar job´s Map/Reduce mediante archivos ejecutables o scripts.

3. Requerimientos

En esta sección se enumeran los requerimientos que debió cumplir el entorno de automatización brindando una breve descripción de los mismos. En las subsiguientes secciones se ahonda en las mismas especificando en cada caso el funcionamiento del mismo.

3.1 Requerimiento 1: Arquitectura Distribuida y Extensible. Portabilidad. Escalabilidad.

Debido a la variabilidad de ambientes a la cual la plataforma debe de ser portable (entorno pc standalone, clúster y cloud), se requiere una arquitectura distribuida y extensible que sea escalable a estos entornos.

Por este motivo se debe diseñar una arquitectura de cliente-servidor con responsabilidades bien definidas, que permita escalar a distintas plataformas y que asimismo permita la interacción con distintas interfaces según el entorno.

3.2 Requerimiento 2: Streaming

Debido a que los archivos que es necesario manipular para realizar procesos como la alineación de datos, pueden ser son del orden de varios gigabytes o incluso terabytes, es de vital importancia que la herramienta permita distribuir la carga y así mismo permita el streaming de datos. Por este motivo, una de sus prestaciones será la de aceptar que las librería externas que manejen streaming de datos, retornen la información en streaming, para que luego, subsecuentes pasos de flujos puedan manejar la información o bien, en streaming o recolectando la misma hasta obtener el consolidado.

3.3 Requerimiento 3: Pausa, Edición y Retoma

Debido a lo pesados en memoria y en cómputo de los procesos y la necesidad de re trabajo que muchas veces el científico se ve obligado a enfrentar, se requiere brindar un entorno que permita que ante la detección de errores en el flujo, se pueda proceder a la pausa del mismo, la edición de parámetros, y la re ejecución a partir de un nuevo punto del flujo ya ejecutado.

3.4 Requerimiento 4: Almacenamiento

Surgido de la necesidad de ejecución de la plataforma en entornos disímiles como cloud, clúster y standalone, se requiere que la plataforma permita un manejo desacoplado de archivos por medio de wrappers que tornen transparente dicha gestión.

3.5 Requerimiento 5: Interfaz Gráfica

Para evitar la utilización de consolas de texto o configuración de archivos con formatos específicos, se requiere una interfaz gráfica. La misma deberá permitir un uso fácil e intuitivo para usuarios que no tengan conocimientos informáticos.

3.6 Requerimiento 6: Entorno Clúster - Hadoop

Los flujos de trabajo asociados a la bioinformática comúnmente se componen por procesos pesados o que requieren un elevado costo computacional. Como muchos de estos procesos se ejecutan en PC´s de escritorio o entornos locales, se requiere opcionalmente la creación de un entorno que permita ejecuciones sobre Hadoop.

Por este motivo, la inclusión de Hadoop al motor de workflow del entorno de automatización brindaría una agilización y reducción considerable de los tiempos de ejecución [Ref24].

3.7 XML Fácilmente extensible

Debido a la variabilidad de datos y herramientas con los cuales deberá interactuar el software a implementar, se requiere el diseño de un XML fácilmente extensible que permita incluir de forma transparente, nueva metadata para modelar las necesidades que vayan surgiendo.

4. Software implementado

Los requerimientos principales expuestos por el cliente fueron por un lado el de implementar una solución que permitiera realizar etapas de ejecución, permitiendo pausar, editar y retomar uno o más trabajos que estuvieran en ejecución. Por otro lado, también se debería permitir la transferencia de datos en formato de streaming entre dos nodos consecutivos.

Como opción se presentó la posibilidad de implementar una extensión de alguno de los software´s de workflow mencionados, tales como Trident Workflow o Taverna Workflow.

Teniendo en cuenta las opciones anteriormente presentadas y con el requerimiento de obtener un producto funcional para una plataforma con sistema operativo Microsoft, se descartó la posibilidad de extender Taverna Workflow pues tal como se mencionó, el mismo está desarrollado en el lenguaje Java.

El software Trident Workflow en cambio, no cumple con ninguno de los requisitos planteados por el cliente. Igualmente, teniendo una noción de la arquitectura de dicho software, se decidió que hacer una adaptación del mismo podría implicar una reestructura compleja de la arquitectura.

Por estos motivos planteados, se descartó la idea de extender un software de workflow existente.

Luego de esta decisión inicial, partiendo de la base de que se debería implementar un software propietario, se consideró utilizar una especificación de lenguaje de workflows existente.

Nuevamente, se descartó la especificación de YAWL por tratarse ésta de una implementación con un motor JAVA.

Como se mencionó anteriormente, WS-BPEL cuenta con una implementación con motor sobre plataforma Microsoft y tecnología .NET llamado BizTalk Server. Dicha propuesta fue presentada como posibilidad pero rechazada por el cliente, argumentando éste disconformidad con la forma de procesamiento de datos y generación de log´s al momento de ejecución.

Debido a las razones previamente expuestas, se decidió implementar un software de workflow propietario, manteniendo el mismo una especificación adaptada a las necesidades dadas, pero partiendo de la base de conocimiento obtenida tras el análisis de los productos y herramientas anteriores.

4.1 Arquitectura y Diseño

Para el diseño del servidor se buscó llegar a una solución flexible, fácilmente extensible, que hiciera uso de buenas prácticas y patrones de diseño.

En primer lugar, como todos los nodos siguen estrategias de ejecución distintas, se utilizó el patrón strategy para modelar los mismos.

Por otro lado, debido a que todos los nodos deben seguir un comportamiento básico, más allá de los comportamientos específicos que los diferencian, se siguió para su modelado el patrón template. De esta forma, se creó una clase genérica que permite que todos los nodos sigan el mismo algoritmo aunque presenten comportamientos distintos.

Todos los nodos presentan el siguiente comportamiento:

```
   mientras (1)

      Mensaje m = ReciboMensajeColaNodo()

      Si mensMePertenece(m) && estoyEnEstadoCorrecto(m)

           Si (ultimoIdMensaje<m.Id)

                cargoParametros(m)

                ultimoIdMensaje=m.Id

                caso (m.tipo):

                     PAUSE
                     procesarPause()
                     CONTINUE
                     procesarContinue()
                     RESUME
                     procesarResume()
                     STREAMING
                     procesarStreaming()

                fin caso
           sino
                retornoMensajeColaNodo(m)
           fin si
      sino
           retornoMensajeColaNodo(m)
      fin si

fin mientras
```

Comportamiento nodo

Asimismo, los wrappers y servicios que se pueden invocar desde los nodos de tipo Task, se implementaron mediante el patrón Factory, para de esta forma, independizar la implementación de las clases que la utilizan.

4.1.1 Vista de Componentes

Con el siguiente diagrama presentamos los componentes de los que consta la solución Imagen 9:

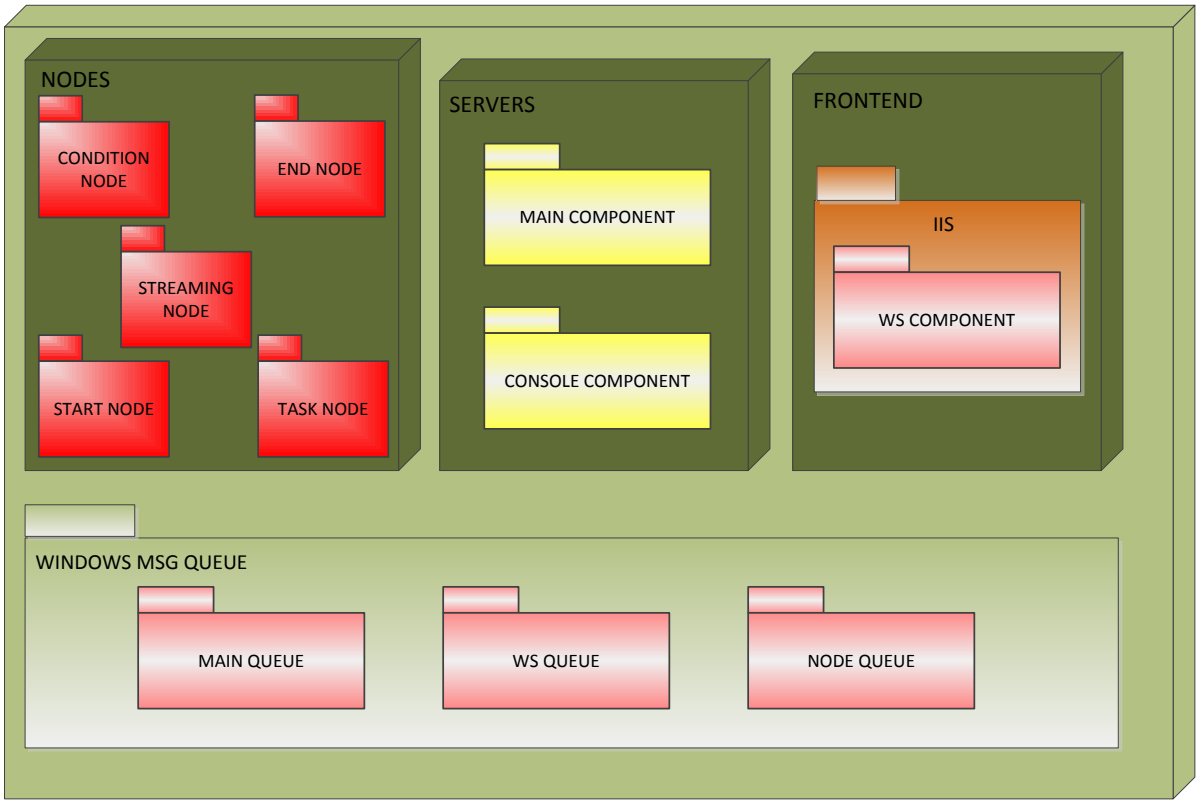

Imagen 9 - Vista de Componentes

El Main component actúa de servidor del motor de workflow, dicho componente es el responsable de mantener las estructuras, realizar el procesamientos y recibir y enviar los comandos a los demás componentes del motor.

Los nodos son de distinto tipo y permiten modelar los distintos procesos a ejecutarse.

Windows Message Queue es un servicio de Windows que permite implementar colas de mensajes. Existen tres tipos de colas de mensajes para la solución, cada una de estas colas es escuchada por un componente distinto.

Console Component es el resposable de recibir comandos a través de la consola y enviarlos al Main component.

GUI Component es la interfaz gráfica de la solución, permite enviar comando al motor del workflow, modelar procesos y visualizar la ejecución de los mismos.

IIS es el servidor sobre el cual corre el componente GUI.

Por otro lado, La instalación del entorno de automatización se realizó separando por un lado el motor de workflow que contiene los componentes nodo, main y console, cada uno con interfaces claramente definidas, y por el otro, el cliente GUI que corre sobre el servidor de Microsoft IIS.

4.1.2 Vista de Deployment

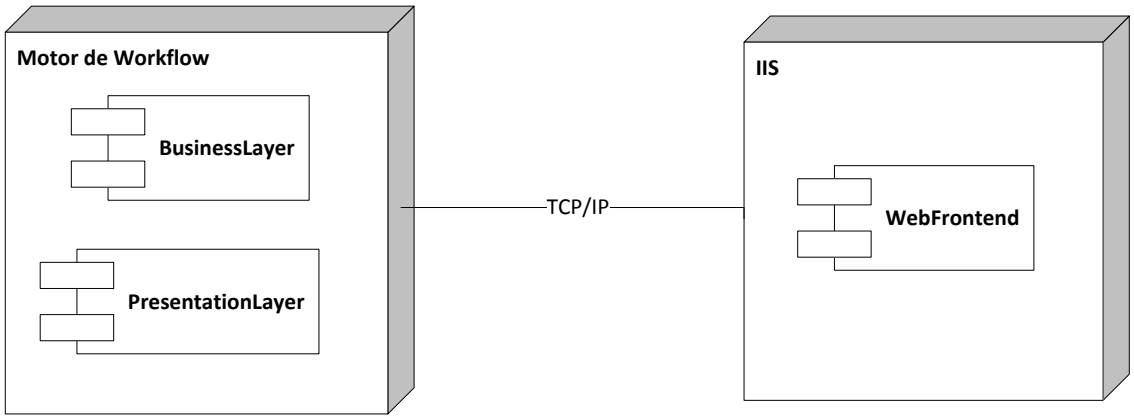

Imagen 10 - Vista de Deployment

La Imagen 9, presenta la vista de componentes. Dichos componentes fueron distribuidos según la Imagen 10, vista de Deployment, de la siguiente forma:

- Los nodos forman parte de la business layer y se encuentran en el motor de Workflow
- El Console Component se encuentra como presentation layer
- El Main Componente forma parte de la business layer
- Las colas de mensaje se encuentran formando parte del motor de workflow y permiten la comunicación entre componentes.
- El WS Component forma parte del Web Frontend.

A continuación se describen los componentes que constituyen al mismo.

4.1.3 Diseño del Motor de Workflow

El motor de workflow se compone de un componente principal (Main Component) que se encarga de la creación e inicialización de las estructuras básicas y de un componente dedicado a la recepción de comandos por consola (línea de comandos).

Ante la carga y comienzo de un nuevo proceso el motor de workflow, desde este componente principal se encarga de la creación de todos los componentes nodo que componen el workflow, la carga de parámetros y de enviar la señal de comienzo.

NODOS

Cada nodo de un nuevo proceso se crea con sus parámetros de entrada, un identificador único por nodo y un identificador único por proceso. Los nodos poseen estados los cuales son los siguientes:

- **WAITING**
  Estado con el que se crean los nodos, cuando en este estado, los mismos se encuentran esperando un mensaje con parámetros de entrada o una señal de comienzo para iniciar ejecución.

- **ACTIVE**
  Estado en el que se encuentra un nodo cuando se encuentra ejecutando. Según el tipo de nodo, la ejecución puede consistir en la evaluación de una condición, la invocación a un servicio externo o la ejecución en streaming.

- **PAUSED**
  Estado en el que se encuentran los nodos cuando el servidor recibe la señal de pausa de un flujo. En este estado los nodos permanecen hasta que reciben la señal de continuar.

- **FINISHED**
  Estado en el que se encuentra un nodo cuando finaliza su ejecución, es decir, cuando el mismo ya transitó por los estados *waiting*, *active* y *paused*. Un nodo puede volver a un estado anterior cuando el flujo se encuentra pausado y el nodo recibe la señal de continuar.

A continuación se presenta un diagrama con los estados y mensajes posibles con sus transiciones Imagen 11:

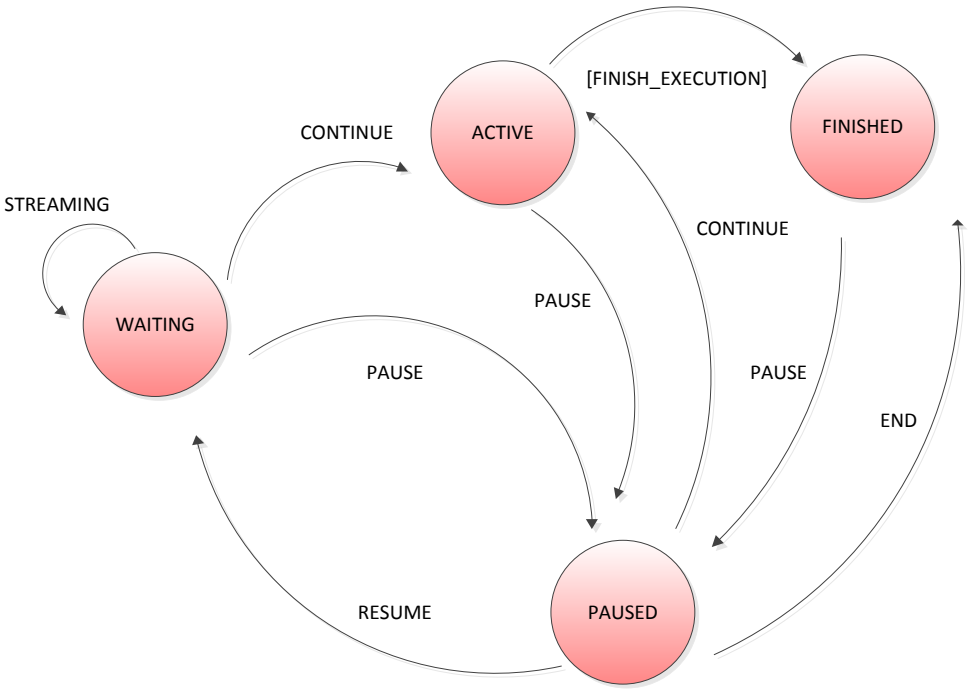

Imagen 11 - Máquina de Estados de los Nodos

Los mensajes que reciben los nodos son enviados mediante colas de mensajes de Windows desde el nodo anterior del flujo o desde el hilo principal del servidor.

COLAS DE MENSAJE

A nivel del motor de Workflow, se manejan tres colas de mensajes de Windows, las cuales son las siguientes:

- **Main Queue**
  La cola MainQueue es escuchada por el hilo principal del servidor. Mediante esta cola se cargan, crean, pausan y retoman los flujos de trabajo que ejecuta el servidor ante solicitudes de la interfaz de usuario consola y se notifica al hilo principal de la finalización de la ejecución de un nodo. De esta forma, el hilo principal lleva un registro de los procesos ejecutando y del estado de los nodos que componen el mismo.

Esta cola recibe un tipo de mensaje específico denominado MessageMain y que permite modelar toda la información necesaria para la ejecución: Resultado, StackTrace y Mensaje de la Excepción. Asimismo, este tipo de mensaje hereda de un tipo de mensaje genérico nombrado "GenericMessage" que se describe más adelante.

- **Node Queue**
  La cola NodeQueue es escuchada por cada nodo que compone el flujo de trabajo y se utiliza para la comunicación con los mismos. Los mensajes que reciben los nodos pueden ser enviados por el hilo principal ante una solicitud de inicio, pausa o retoma o por parte del nodo anterior del flujo.

  Esta cola recibe un tipo de mensaje específico denominado MessageNode que permite modelar toda la información necesaria para la ejecución. Asimismo, el tipo de mensaje MessageNode hereda de un tipo de mensaje genérico que se describe más adelante.

- **WS Queue**
  La cola de mensajes Ws Queue se utiliza para la comunicación entre la interfaz gráfica y el hilo principal del servidor. Esta cola es escuchada por la interfaz gráfica y es utilizada por el servidor para comunicar el estado de los nodos del proceso que se está monitoreando desde la interfaz gráfica: por ejecutar, finalizado o ejecutando.

A continuación se muestra gráficamente la interacción entre los distintos componentes de la solución:

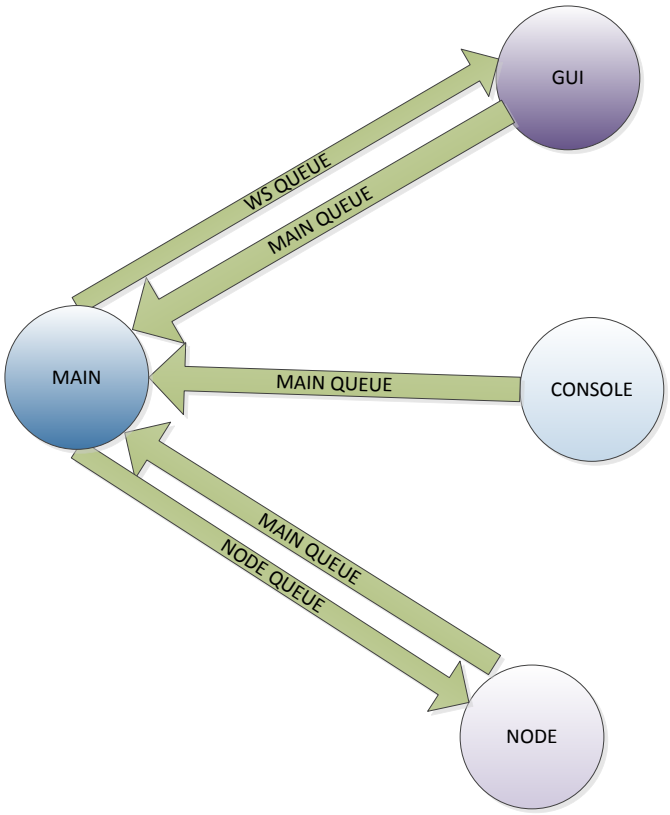

Imagen 12 – Interacción entre colas de mensajes

En la imagen 12 se pueden visualizar los tres componentes principales de la solución: el módulo Main, el módulo Console y los módulos Node. Los mismos, interactúan a través de las colas de mensaje Main Queue, Node Queue y WS Queue según el diagrama anterior.

PROTOCOLO DE MENSAJERÍA

Todos los mensajes intercambiados por la solución heredan de un tipo de mensaje genérico denominado GenericMessage el cual mantiene la siguiente información:

- **Id**
  Identificador del nodo destinatario del mensaje, permite discernir al nodo si debe o no procesar el mensaje.

- **ProcessId**
  Identificador del proceso destinatario del mensaje, permite en conjunto al Id discernir al nodo si debe o no procesar el mensaje.

- **Origin**
  Origen del mensaje, es el proceso o nodo que lo envío.

- **MsgType**
  Tipo de mensaje: PAUSE, RESUME, END, STOP, CONTINUE, EXIT, LOAD, STATUS_REQUEST, STREAMING. Permite saber la acción deseada. Se explican más adelante.

- **MsgId**
  Identificador del mensaje, permite mantener la secuencia descartando mensajes viejos. Este identificador es especialmente importante ante mensajes viejos que quedan luego de instrucciones de pausa.

- **Parameters**
  Lista de parámetros de entrada que debe procesar el nodo o el hilo principal del servidor.

  Este listado es utilizado para enviarle al nodo resultados obtenidos en instancias anteriores. Cuando estos resultados son simples como números, booleanos o strings cortos, se pueden enviar directamente.

  En el caso de archivos grandes se puede enviar la ruta en disco en donde se encuentra el archivo a cargar.

  En el caso del servidor, se utiliza este listado para la carga o inicio de procesos y para la edición de parámetros ante un pedido de pausa, edición y retoma.

  Los parámetros contienen la siguiente información:

  - Name: Nombre del campo

  - PamValue: Valor del campo

  - Type: tipo del campo (boolean, string, number,etc..)

- Action: Acción a ejecutar sobre el parámetro:

    - New: el parámetro es nuevo y se debe agregar.
    - Replace: el parámetro ya existe y se debe sustituir. Este comando se utiliza ante la instrucción de edición del comando pausa, edición y retoma.
    - Partial: si el parámetro proviene de un streaming de datos y se debe concatenar al final del ya almacenado.
    - Finish: si el valor contenido finaliza el parámetro de entrada. Este es el caso de cuando se envía el último dato en streaming.

- MaxIndex: Dato utilizado cuando se está ante streaming de parámetros de entrada, este dato es mantenido a nivel del nodo para conocer a qué nivel debe realizar la concatenación de parámetros.

EJECUCIÓN DE UN PROCESO

El proceso para que ocurra una acción comienza desde uno de los componentes UI: console o GUI. Uno de dichos componentes recibe un comando del usuario a realizar una acción. Esa acción es transmitida mediante un mensaje depositado en la cola Main Queue al componente Main. El componente Main retransmite el mensaje a los nodos involucrados a través de la cola de mensajes Node. Cuando un nodo completa la acción notifica al componente Main, el cual actúa como servidor a través de la cola de mensajes Main Queue. En el caso de la GUI que tiene que reportar el estatus de los procesos, la misma recibe mensajes a través de la cola de mensajes WS Queue.

El siguiente diagrama explica el flujo que sigue un comando cuando el mismo es originado desde el componente GUI:

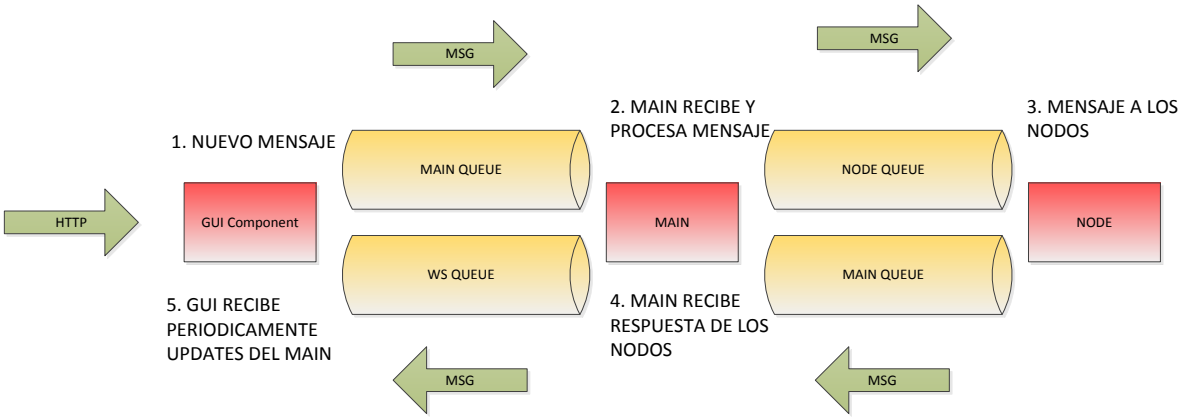

Imagen 13– Proceso de Ejecución desde componente GUI

En la imagen 13 se ve como el comando se origina en el componente GUI, a través de la cola de mensajes Main es recibido por el componente Main y dicho componente, lo procesa. El procesamiento en el componente Main implica la actualización de estructuras internas, la generación de un mensaje para los nodos y el envío de dicho mensaje a través de la cola de mensajes Node Queue. Asimismo, el componente GUI envía al componente Main solicitudes de actualización de estatus del proceso, las cuales son contestadas a través de la cola de mensajes WS Queue.

El siguiente diagrama explica el flujo que sigue un comando cuando el mismo es originado desde el componente Console:

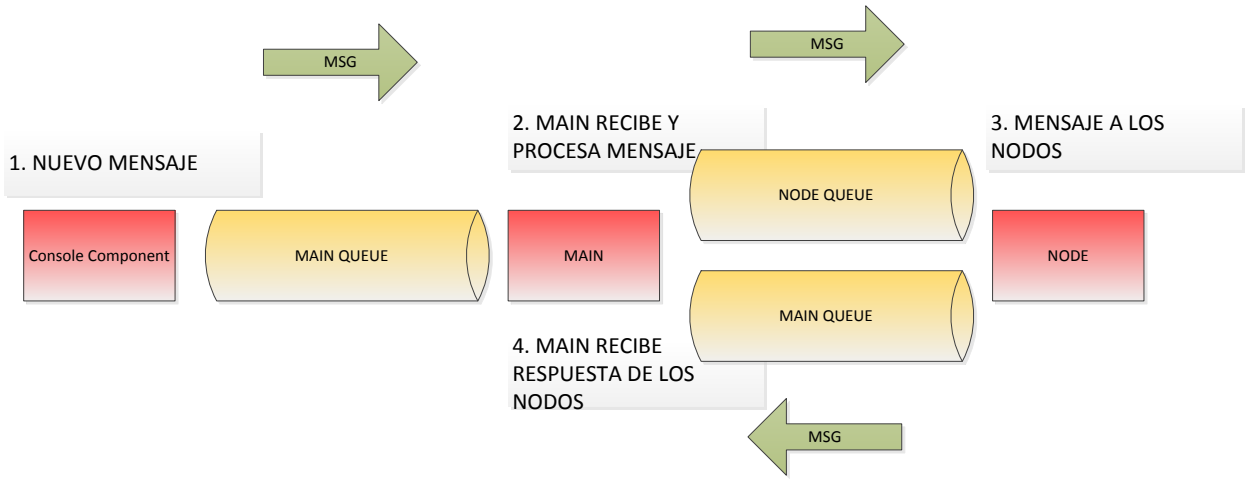

Imagen 14 – Proceso de Ejecución desde componente GUI

En la Imagen 14 se puede apreciar como el proceso es el mismo, exceptuando por el envío de mensajes de solicitud de estatus y su contestación.

4.1.4 Interfaz Gráfica

COMUNICACIÓN ENTRE INSTANCIAS

La interfaz Gráfica fue desarrollada para ser presentada en un ambiente Web, permitiendo de esta manera la independencia de plataforma o sistema operativo a utilizar.

Esta interfaz permite al usuario tanto la creación como la ejecución de estructuras de workflow, pudiendo también monitorear el estado de la ejecución en tiempo real.

Para poder establecer la comunicación entre la interfaz Web y el servidor de aplicación, se debió crear un método de comunicación basado en Web Services. Esto se debe a que la capa de negocios no cuenta con la capacidad de comunicarse directamente con la interfaz web del usuario. A su vez, se debió implementar una comunicación de los Web Services con la capa de negocio mediante colas de mensajes. El esquema de comunicación se detalla en Imagen 15.

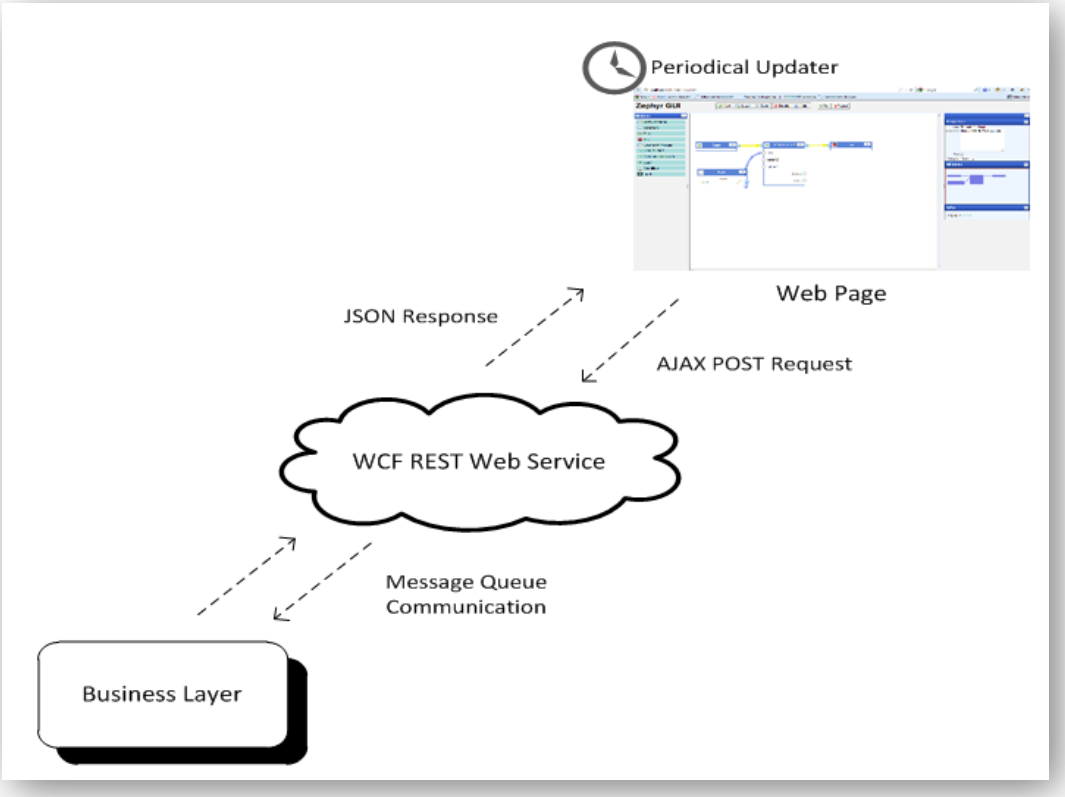

Imagen 15 - Comunicación Interfaz Gráfica

WEB PAGE – NEW / LOAD / SAVE WORKFLOW

Cada usuario cuenta con la posibilidad de crear, persistir o cargar su propia estructura de workflow desde el entorno Web. Para poder ofrecer esta funcionalidad, se implementaron una serie de servicios en un Web Service REST sobre la plataforma WCF (Windows Communication Foundation).

Antes de poder ejecutar una nueva estructura de workflow, ésta debe ser previamente almacenada. Al realizar dicha acción, el navegador invoca al servicio encargado de la persistencia mediante un llamado AJAX. Este servicio es invocado de la siguiente manera:

[url]/[nombre del WService]/save/structure

El pedido AJAX generado, contiene la estructura del workflow creado por el usuario en formato JSON y es recibido en este formato por el Web Service. Para poder establecer una relación entre el formato JSON y la estructura XML, fue necesario implementar un traductor, manejado por la clase “Translator”.

Ejemplos de estructuras JSON junto a sus correspondientes archivos XML pueden ser encontrados en la sección 9.2 - Estructuras.

Una vez persistida la estructura del workflow, el usuario contará con la posibilidad de ejecutar o pausar dicha ejecución. Estas funcionalidades se encuentran disponibles mediante la invocación de los siguientes servicios:

[url]/[nombre del WService]/run

y

[url]/[nombre del WService]/pause

respectivamente. En ambos casos se realiza un llamado AJAX que contiene el identificador del proceso de workflow a ser tratado.

WEB PAGE – WORKFLOW STATUS

Para que el usuario pudiera contar con un feedback en tiempo real del estado de ejecución del workflow, se debió implementar un sistema de consulta que se invocara repetitivamente cada cierto período de tiempo.

Esta funcionalidad se implementó utilizando la tecnología AJAX, con la función provista por la misma “PeriodicalUpdater”. Un PeriodicalUpdater se mantiene en ejecución con un período de tiempo establecido (en nuestro caso, cada 1 segundo) hasta recibir una señal de terminación.

Una vez ejecutado el sistema de workflow, se mantiene una invocación constante al servicio “getStatus” del Web Service involucrado hasta que en el flujo se alcance un nodo de tipo fin o que el usuario decida pausar o cancelar la ejecución.

La invocación para obtener el status de un flujo de trabajo se debe realizar de la siguiente manera:

[url]/[nombre del WService]/getStatus/[workflowID]

La invocación de este servicio se realiza mediante un llamado de tipo POST, manejando un timeout de respuesta de hasta 3000 milisegundos.

Como respuesta, el servicio retorna una estructura representada en formato JSON que contiene la información de cada nodo, ya sea si concluyó su ejecución, está en proceso o si aún no se ha ejecutado.

Un ejemplo de respuesta del estado de un servicio se da en la sección 9.2 - Estructuras.

TRANSLATOR

Tal como se describió en la sección 4.1.2 - Manejo de estructuras sobre manejo de estructuras por parte de la capa de negocio, se describe en esta sección el manejo de estructuras JSON utilizadas por la interfaz Web.

Considerando la interfaz Web, los nodos y asociaciones representados, deben manejar una metadata más compleja que los representados en XML. Al momento de procesar la información del archivo XML, no importa la posición del nodo en el canvas, el tipo de arco con el que se representa un link o el color de dicho arco. Es por este motivo que la clase encargada de realizar la traducción, descarte información innecesaria.

Para realizar una analogía con la sección 4.1.1 - Modelado de XML, un nodo con las siguientes características:

- Id de nodo = 2
- Nodo que invoca un Web Service (en este caso un servicio de la NCBI para obtener taxonomías a partir de nombres) “NCBI FETCHTAXON”.
- Invocación del Web Service con parámetro de tipo string y valor “cat”
- Nodo invocador con id = 1
- Siguiente nodo a invocar con id = 4

Genera una estructura como la siguiente:

```
[{"working":"{\"modules\":[{\"config\":{\"position\":[251,83],\"
xtype\":\"WireIt.InOutContainer\"},\"name\":\"NCBIEUTILS\",\"val
ue\":{}},{\"config\":{\"position\":[473,80],\"xtype\":\"WireIt.I
nOutContainer\"},\"name\":\"End\",\"value\":{}},{\"config\":{\"p
osition\":[32,162]},\"name\":\"input\",\"value\":{\"input\":{\"i
nputParams\":{\"typeInvite\":\"inputname\",\"value\":\"cat\"},\"
type\":\"string\"}}},{\"config\":{\"position\":[38,85],\"xtype\"
:\"WireIt.InOutContainer\"},\"name\":\"Start\",\"value\":{}}],\"
wires\":[{\"src\":{\"moduleId\":2,\"terminal\":\"SOURCES\"},\"tg
t\":{\"moduleId\":4,\"terminal\":\"FOLLOWUPS\"}},{\"src\":{\"mod
uleId\":1,\"terminal\":\"out\"},\"tgt\":{\"moduleId\":2,\"termin
al\":\"term\"}},{\"src\":{\"moduleId\":3,\"terminal\":\"SOURCES\
"},\"tgt\":{\"moduleId\":2,\"terminal\":\"FOLLOWUPS\"}}]}"}]
```


Estructura JSON

4.2 Diseño de la solución

El diseño de la solución se puede apreciar en la Imagen16:

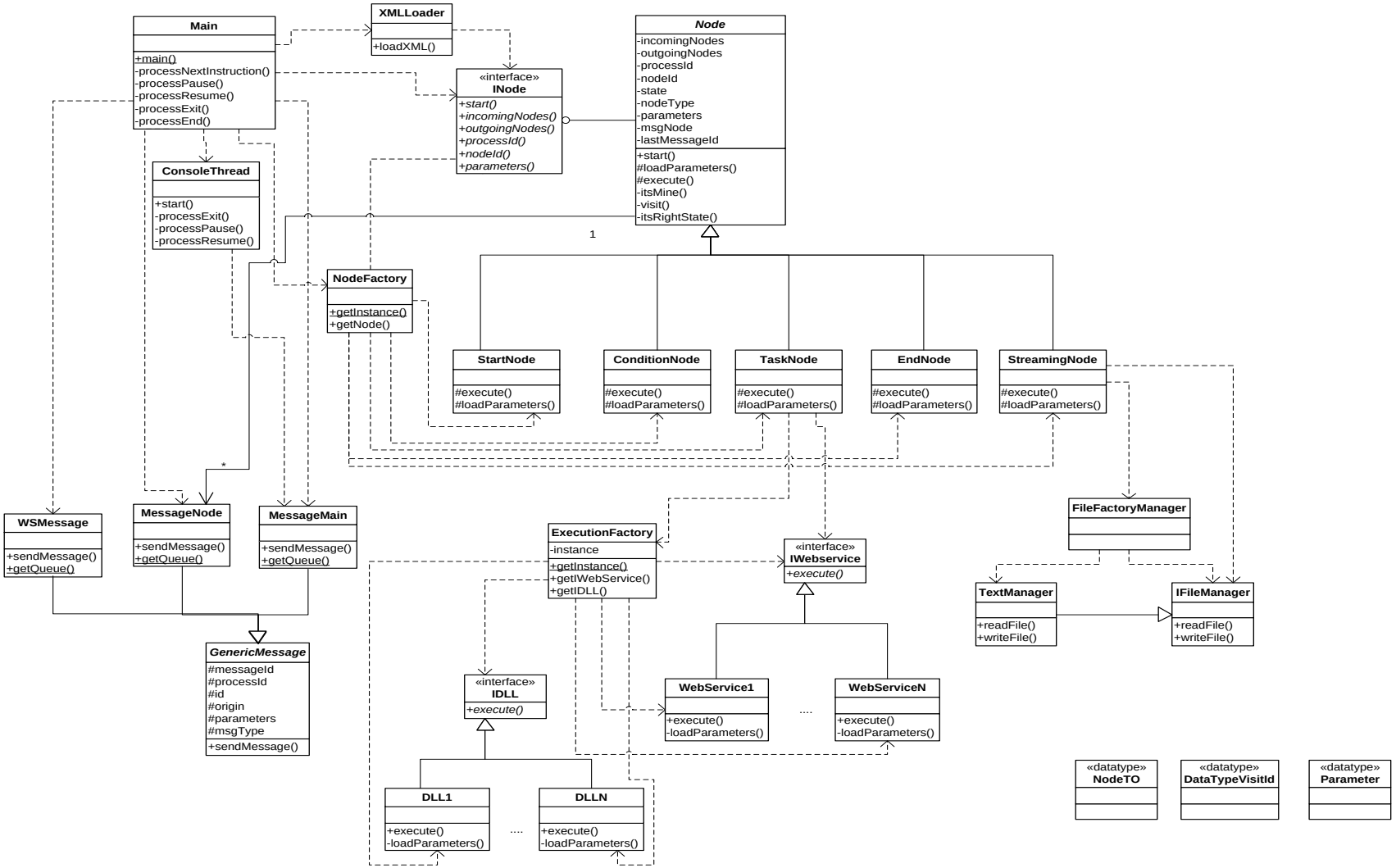

Imagen 16 - Diagrama de Clases

4.3 Implementación

Se desarrolló una herramienta informática que mediante la aplicación de algoritmos bioinformáticos a distintos datos biológicos, permita realizar diversos procesos científicos comunes al post-procesamiento de las secuencias de ADN, simplificando la parametrización y ejecución de dichos procesos en diversos entornos, que abarcan los modelos de ejecución de validaciones en Computadoras Personales hasta teóricamente la distribución y ejecución en clúster y/o entornos de Cloud Computing.

4.3.1 Tecnología

Para la construcción del servidor se utilizó el lenguaje de programación C# .NET sobre Microsoft © .NET Framework 4, WCF y colas de mensaje de Windows.

Por otro lado, para la construcción de la interfaz gráfica se utilizó JavaScript (puro y modelado de objetos con JSON [Glo 7.1.7]), HTML y servicios REST sobre Microsoft © .NET Framework 4.

Como apoyo para la interfaz web se utilizó la librería javascript Wire-IT en su versión 0.5.0. [Ref18]

La plataforma de desarrollo fue Microsoft © Visual Studio 2010.

La herramienta de control de cambios utilizada fue el SVN ofrecido por Google Code.

4.4 Funcionalidades

A continuación se presentan las funcionalidades contenidas en el software implementado.

4.4.1 Modelado de XML

Al contarse con una persistencia y carga de estructuras en archivos con formato XML, debió implementarse una lógica de conversión objeto-archivo. De esta manera, no solo se tomaron en cuenta los nodos del workflow, sino también los links entre ellos.

Cabe considerar que cada nodo presenta funcionalidades diferenciadas según su tipo, pero con una estructura general compartida, tal como se puede apreciar en la Imagen17.

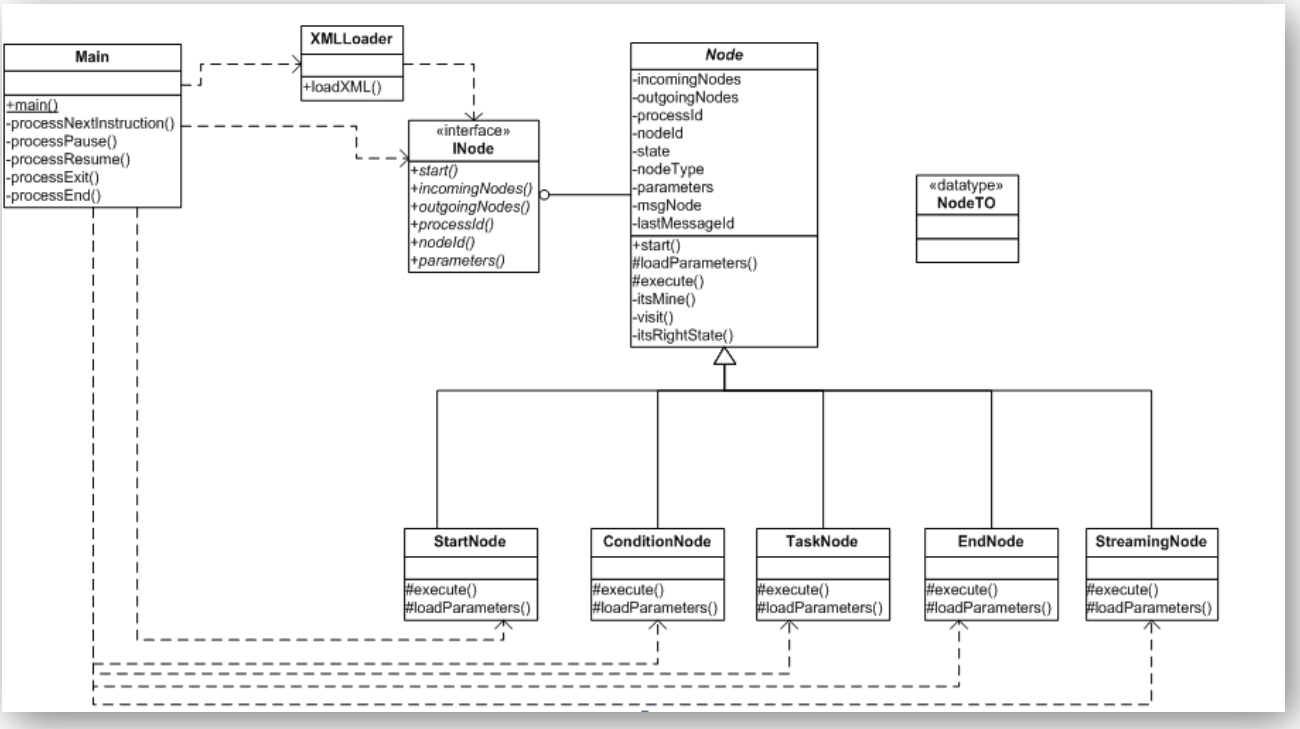

Imagen 17 - Diseño Nodos

Tanto para la escritura como lectura de la información de cada nodo, se debió crear una estructura serializable implementada con un DataType nombrado NodeTO. Dicho DataType mantiene la información de cada nodo en particular, como lo es el identificador del nodo, tipo de nodo, lista de enlaces entrantes, lista de enlaces salientes y parámetros cargados. Estos parámetros se describen en las siguientes secciones.

Para un nodo con las siguientes características:

- Identificador del nodo actual, con valor 2
- Nodo que invoca un Servicio Web (en este caso un servicio de la NCBI para obtener taxonomías a partir de nombres) con valor “NCBI FETCHTAXON”.
- Invocación del Web Service con parámetro de nombre term, tipo string y valor “cat”
- Nodo predecesor con identificador con valor 1
- Siguiente nodo a procesar con identificador con valor 4

se genera un fragmento XML con el formato representado en la siguiente tabla [Tabla3].

XML

```
<NodeTO>
     <id>2</id>
     <nodeType>TASK_NODE</nodeType>
     <parameters>
       <Parameter>
         <name>WS_TYPE</name>
         <pamValue>NCBI FETCHTAXON</pamValue>
         <type />
         <action>NEW</action>
       </Parameter>
       <Parameter>
         <name>term</name>
         <pamValue>cat</pamValue>
         <type>string</type>
         <action>NEW</action>
       </Parameter>
     </parameters>
     <incomingNodes>
       <DataTypeVisitId>
         <Id>1</Id>
         <visited>false</visited>
       </DataTypeVisitId>
     </incomingNodes>
     <outgoingNodes>
       <int>4</int>
     </outgoingNodes>
   </NodeTO>
```

Tabla 3- XML Nodo

El encabezado del archivo generado o leído cuenta con información general de XML como lo es el encode, versión y atributos, seguido por una etiqueta <_nodes>. Luego se presenta la lista de nodos pertenecientes a la estructura de workflow generada.
El modelado de un workflow entonces, queda representada como una secuencia de nodos sin necesidad de respetar un orden en particular, pues el mismo es establecido con los id´s de nodos y sus respectivas referencias de entrada y salida.
Cabe destacar que los enlaces entre los nodos, junto a sus respectivas propiedades, quedan representados dentro de dichos nodos y no como estructuras independientes.

Una estructura de workflow completa es presentada en la sección 9.2 - Estructuras.

4.4.2 Manejo de estructuras

Ya sea para ejecutar, cargar o persistir un modelo de flujo de trabajo, se deberá contar con una estructura XML asociada a dicho modelo. Tal como se mencionó en la sección 4.1.1, se debió implementar una lógica de conversión objeto-archivo que pudiera manejar este mapeo.

La clase encargada de esta función fue nombrada XMLLoader.

A su vez, la librería javascript utilizada en la interfaz web para la creación de bloques, presenta un formato de representación propietario manejado con JSON.

Para poder trabajar entonces con flujos de trabajo modelados con la herramienta web, se debió implementar un traductor. El mismo debe encargarse de convertir las estructuras formadas en JSON a nuestro formato XML y viceversa Imagen18.

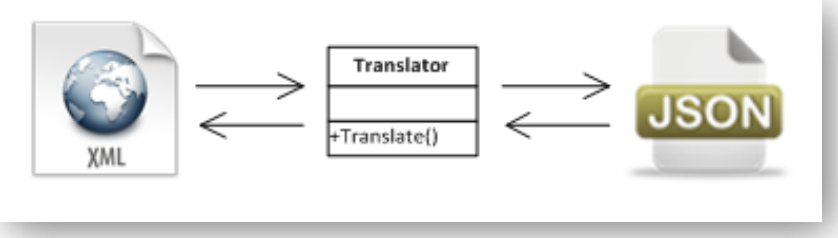

Imagen 18 - Traductor

El modelado de los archivos JSON se presenta en la sección referente a la arquitectura relacionada a la interfaz gráfica 4.2 - Arquitectura.

En la sección 9.2 - Estructuras se detallan los archivos XML y JSON creados para un flujo simple como el presentado en la Imagen19.

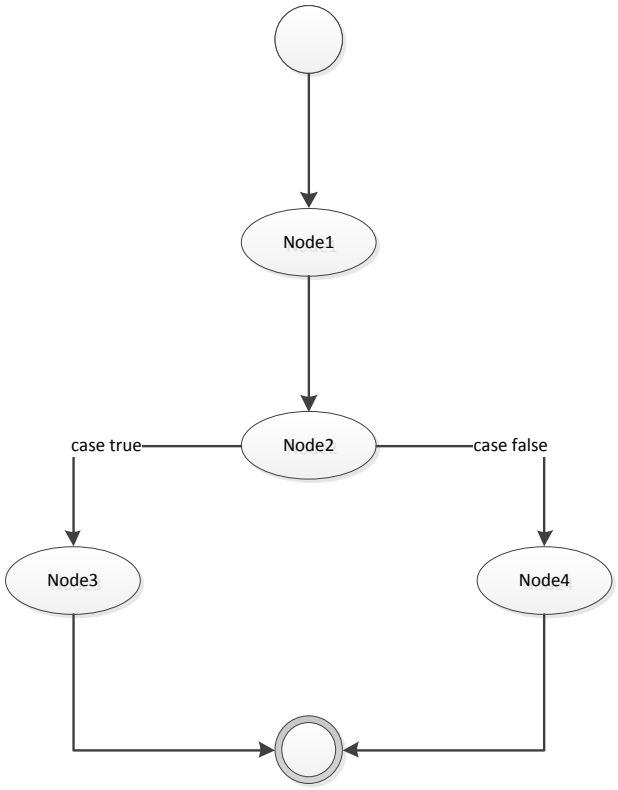

Imagen 19 - Modelo de Flujo

4.4.3 Tipos de Nodo

Los tipos de nodo modelados por el motor de workflow son los siguientes:

- **ConditionNode**
  Este tipo de nodo permite modelar condiciones como ser comparaciones de mayor que, menor que, igual que, mayor o igual que, menor o igual que y distinto. Recibe dos parámetros IComparable, la condición que se desea evaluar y el nodo al cual transitar en caso de resultado positivo y en caso de resultado negativo (true o false).

- **StartNode**
  Este nodo inicia la ejecución, no posee predecesor y es el que recibe la señal de comienzo por parte del servidor.

- **EndNode**
  Este nodo es el responsable de notificar al servidor de la finalización del flujo.

- **TaskNode**
  Este nodo es el responsable de ejecutar servicios web y librerías dll. Es fácilmente extensible y posee una Factory de servicios y dlls asociada que permite modelar fácilmente los distintos servicios separando la implementación de la lógica del nodo.

- **StreamingNode**
  Este tipo de nodo es el responsable de enviar en streaming datos. En caso de librerías que permitan el envío en streaming, se debe extenderlas de este nodo. Todos los nodos del workflow reciben en streaming pero solo este nodo permite el envío en streaming.
  A este nodo se le debe declarar el parámetro que se desea enviar en streaming.

4.4.4 Ejecución

El servidor permite la ejecución en forma concurrente de procesos. El mismo se encuentra conformado por un hilo principal el cual es responsable de cargar, iniciar, pausar, retomar y finalizar flujos de trabajo. Estas instrucciones se ejecutan mediante la recepción de mensajes.

El servidor atiende los siguientes tipos de mensaje:

- **EXIT**: solicita la finalización del servidor. El resultado de este mensaje es la baja de todos los nodos que se encuentran corriendo y del hilo que atiende la interfaz consola (línea de comando).

- **CONTINUE**: notifica que un nodo de un proceso finalizó. El resultado de este mensaje es la actualización de la estructura que mantiene el estado de los nodos de cada proceso.

- **PAUSE**: solicita la pausa de un flujo de trabajo. El resultado de este mensaje es la pausa del flujo de trabajo solicitado.

- **RESUME**: solicita la retoma y edición de un flujo de trabajo. El resultado de este mensaje es opcionalmente la edición de un parámetro y la retoma del flujo a partir del nodo especificado para el proceso solicitado.

- **END**: notifica la finalización de un flujo de trabajo. El resultado de este mensaje es la actualización de la estructura que mantiene el estado de ejecución de los procesos.

- **LOAD**: solicita la carga y creación de un nuevo flujo de trabajo. El flujo se carga de disco, ingresando como parámetro la ruta bajo la cual se encuentra el mismo. El resultado de la ejecución es la instanciación de los nodos que componen el flujo, la carga en memoria de las estructuras con la información del mismo y el envío de una señal de comienzo (mensaje continue) al nodo start del flujo de trabajo.

- **STATUS_REQUEST**: solicitud enviada por la interfaz gráfica para obtener información de la ejecución de un flujo de trabajo. La respuesta a este tipo de solicitud es el envío de un listado de nodos con el estado de los mismos.

Como se mencionó anteriormente, la ejecución de un nuevo flujo de trabajo se realiza mediante la instrucción load. Dicha instrucción implica la lectura del xml que modela el flujo de trabajo a crear, la creación e inicialización de los nodos cada uno en un nuevo hilo y el envío de un mensaje de comienzo. La inicialización de nodos consiste en la carga de los parámetros de entrada con los que se cuenta al momento, la carga del nodo siguiente, del identificador del proceso y del identificador del nodo. Asimismo, en el caso de nodos de tipo task se carga el servicio que se desea ejecutar o ante nodos condición, la condición que se desea evaluar.

4.4.5 Streaming

La funcionalidad de streaming permite al motor de workflow recibir mensajes de manera fraccionada y continua. Esta característica le permite a un nodo ir colectando la salida en streaming de un nodo anterior, permitiendo así al nodo anterior independizarse del comportamiento de la librería en streaming que esté invocando.

La presente funcionalidad se implementó mediante una máquina de estados común a todos los nodos, donde cada nodo en tiempo de ejecución se puede encontrar en un estado distinto. Con este objetivo, se modeló la comunicación entre nodos con mensajes secuenciales y banderas de control. Cada nodo no ejecuta su acción asociada hasta

recibir la totalidad de sus mensajes, evento que se produce cuando recibe una señal de terminación. Por este efecto, nodos anteriores pueden dividir la información e ir enviándola, sin por este motivo, perjudicar la ejecución.

Existen librerías que dado un parámetro de entrada, ejecutan algoritmos que van retornando resultados parciales, en estos casos, la salida en streaming permite que el nodo siguiente pueda manejar estas N salidas.

Para poder desarrollar la funcionalidad de streaming se implementaron las siguientes características:

1. **Mensajes con identificador secuencial**
   Todos los mensajes que se envían poseen un identificador secuencial, un identificador de proceso y un nodo destinatario. Por otro lado, cada nodo lleva un registro del identificador del último mensaje procesado. Luego, por estas características, cada nodo sabe si el mensaje recibido se encuentra dirigido hacia él, y en base a esto, al identificador del nuevo mensaje y el registro del secuencial del último mensaje procesado, pueda evaluar si debe procesar el mensaje o descartarlo.

2. **Último índice recibido para un parámetro de entrada**
   Para cada parámetro con el cual se desea realizar streaming el nodo receptor mantiene el último índice recibido para ese parámetro. Esto le permite poder ir concatenando en orden la información hasta obtener la totalidad del parámetro.

3. **Concatenación de mensajes**
   La concatenación de mensajes permite que parámetros de entrada grandes como ser MultiFasta puedan ser recibidos en etapas sin por ello verse afectado el funcionamiento del flujo.

4. **Envío de parámetros de entrada con xpath**
   Debido a que streaming se encuentra especialmente diseñado para archivos del orden de varios gigabytes, se incluyó un tipo de parámetro ruta que le indica al nodo que debe leer de disco dicho campo.

5. **NodoStreaming**
   Se generó un nodo Sreaming que envía parámetros en etapas. Este tipo de nodo incluye la lógica que se debe incluir en los nodos que manejen servicios con streaming y se creo principalmente como una prueba de concepto.

A fines ilustrativos se realiza un esquema de comunicación entre un nodo emisor de datos en streaming y un nodo receptor cualquiera.

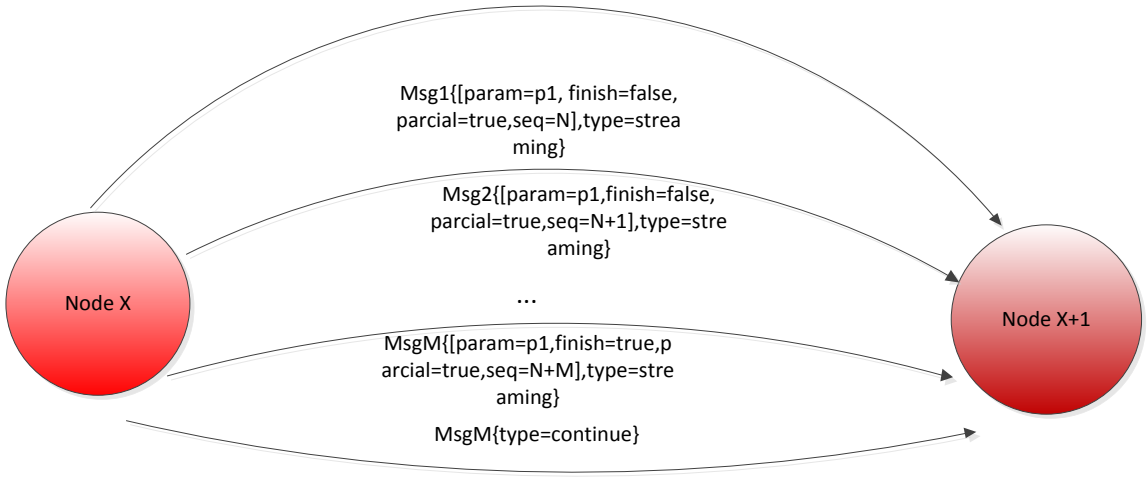

Imagen 20 – Comunicación en Streaming

En la Imagen 20, se puede apreciar la comunicación en streaming entre dos nodos. En dicha imagen se visualiza como el nodo origen le envía al nodo destino un primer mensaje en el que le envía información (parámetros) y le notifica que dicha información le llegará en streaming (type streaming).

Debido a que no todos los parámetros puede que se desee que se envíen en streaming, se especifica en cada caso si la información que contiene el parámetro es parcial o total (partial).

En caso de ser parcial se especifica el número de secuencia para que el armado se realice en orden y se especifica si dicha información es la última que se recibirá del parámetro o si se continuarán recibiendo mensajes para el armado del parámetro (finish).

Cuando el nodo destino recibe la especificación finish para un parámetro que estaba armando y para el cual ya tiene todos los números de secuencia anteriores, lo da como finalizado y evalúa si ya tiene toda la información para ejecutar.

En caso de tenerla, procede a ejecutar, caso contrario, espera sucesivos mensajes que completen los parámetros que le restan para poder ejecutar.

4.4.6 Pausa y Retoma

La funcionalidad de pausa y retoma permite al científico corregir resultados sin que por ello, deba re ejecutar todo el flujo de trabajo completo.

Bajo este requerimiento se ofrecen las siguientes funcionalidades:

1. Pausa del flujo de trabajo.
2. Edición de un parámetro a elección.
3. Retoma del flujo desde cualquier nodo ya ejecutado.

Estas funcionalidades se pueden acceder desde la interfaz de usuario consola (línea de comando).

Cuando un usuario pausa un flujo de trabajo, el hilo que atiende la consola envía un mensaje de pausa al hilo principal, quien a su vez lo propaga a los nodos del flujo.

Los nodos del flujo se encuentran siempre escuchando en la cola de mensajes, cuando reciben este mensaje cambian su estado a PAUSED y vuelven a escuchar de la cola de mensajes.

Cuando el usuario ingresa el comando resume para un determinado flujo de trabajo se le solicita ingrese desde cual nodo desea retomar y si existen parámetros que desee cambiar. En caso desee cambiar un parámetro, se le solicita el nombre del parámetro, el nuevo valor y el tipo.

Esta información ingresada en la retoma, es a su vez enviada al hilo principal que una vez que la recibe, se encarga de avisarle al nodo desde el cual se desea retomar, el nuevo valor del parámetro y que se desea empiece a ejecutar.

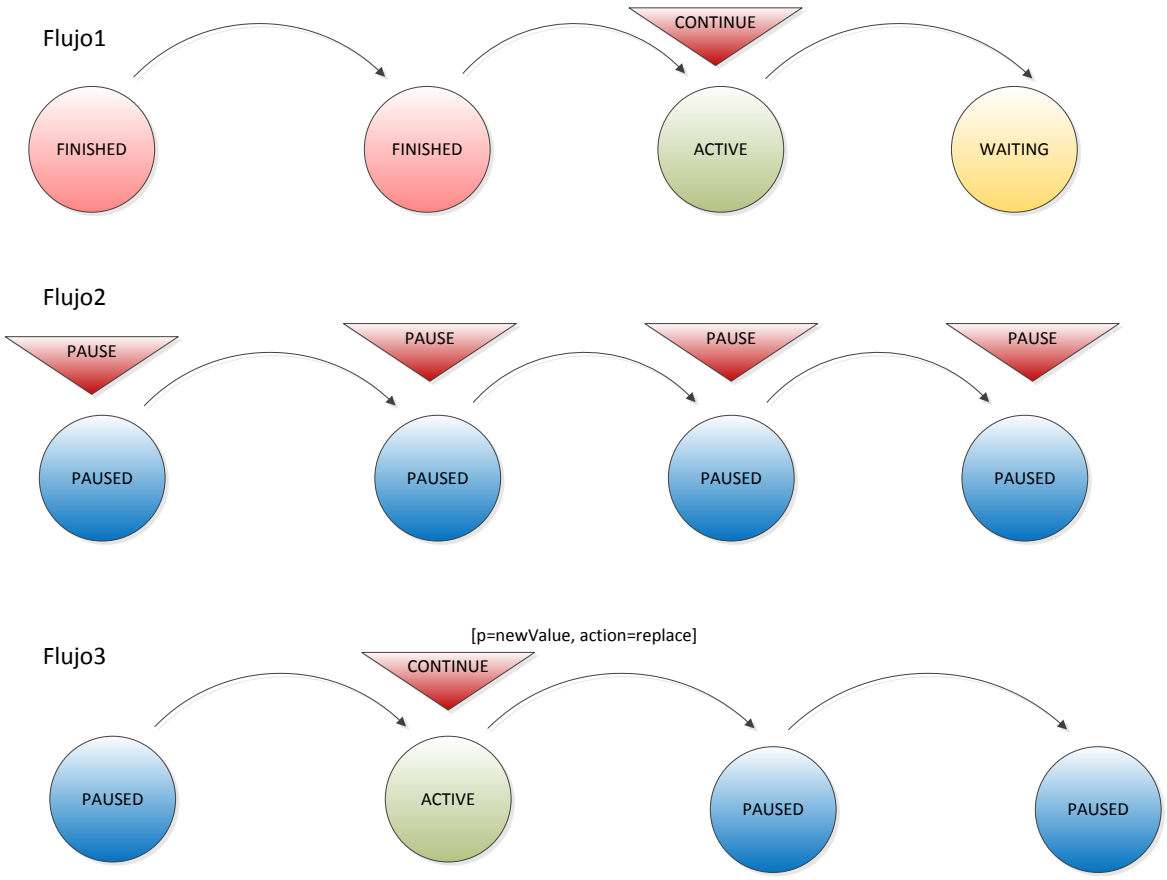

Imagen 21 – Flujo de Pausa, Edición y Retoma

En la Imagen 21 se puede apreciar el flujo de pausa edición y retoma en sus tres estados:

1. Ejecutando
2. Pausado
3. Retoma con Edición

En el primer flujo se tiene a los nodos 1 y 2 finalizados, al nodo 3 ejecutando y al nodo 4 esperando el mensaje de *continue*. El último mensaje enviado en este flujo fue al nodo 3, el cual recibió el mensaje de "*continue*".

En el segundo flujo tenemos a todos los nodos de flujo pausados, todos los nodos recibieron por igual el mensaje de pausa.

En el tercer flujo se tiene una retoma desde el nodo 2. Este nodo recibió el mensaje continue junto a la solicitud de edición de un parámetro por parte del hilo principal.

En este tercer estado se tiene que todos los resultados calculados en la ejecución 1 se mantuvieron, viéndose cambiados a partir del nodo 2 debido al cambio en el parámetro p, el cual produce nuevos resultados en cadena.

Es importante destacar que la implementación de esta funcionalidad requirió que todos los nodos del workflow cumplieran con las siguientes características:

1. Numeración de los mensajes.
2. Envío y recepción en secuencia.
3. Recepción en orden, si se recibe un mensaje anterior al último recibido se descarta el mismo.
4. Corroboración de ser el destinatario del mensaje (por proceso e identificador de nodo).
5. Máquina de estados por nodo. Un nodo solo puede ejecutar cuando se encuentra en un estado válido de ejecución, siendo las acciones permisibles de ser realizadas dependientes del estado.

La numeración de los mensajes fue necesaria para asegurar que la recepción de los mismos fuese en orden. Si un mensaje es anterior al último procesado se descarta. Si un mensaje no es el siguiente al último recibido y es un mensaje de tipo streaming o continue, se vuelve a la cola, en cambio, si el mensaje es de tipo pause o end, se ejecuta aunque se tengan que descartar posteriormente los mensajes con secuencias anteriores.

Debido a que todos los mensajes destinados a los nodos son depositados en una única cola, registrar en cada mensaje el destinatario del mismo. De esta forma, cada nodo conoce su identidad y verifica que el mensaje recibido le fue dirigido. En caso de no ser el destinatario lo devuelve a la cola.

La máquina de estados por nodo fue necesaria para que los nodos únicamente pudieran procesar los mensajes relativos a su estado.

4.4.7 Interfaz de Usuario

El entorno de ejecución de workflows se presenta con dos modalidades de ejecución. Por un lado la consola de comandos, donde el procesamiento se realiza de forma manual y

con el conocimiento previo de los comandos a invocar. Por otro lado, una consola de interfaz gráfica de uso intuitivo y sin necesidad para el usuario de conocimientos informáticos previos.

Consola de Comandos

Una opción para el usuario, es la creación en forma manual de los archivos XML. En este caso el usuario debe de ser especialmente cuidadoso seguir una correcta estructura, respetando la metadata y el flujo del archivo.

Los archivos pueden ser posteriormente cargados para su utilización mediante una consola de comandos con el comando “load”, tal como se muestra en la Imagen22.

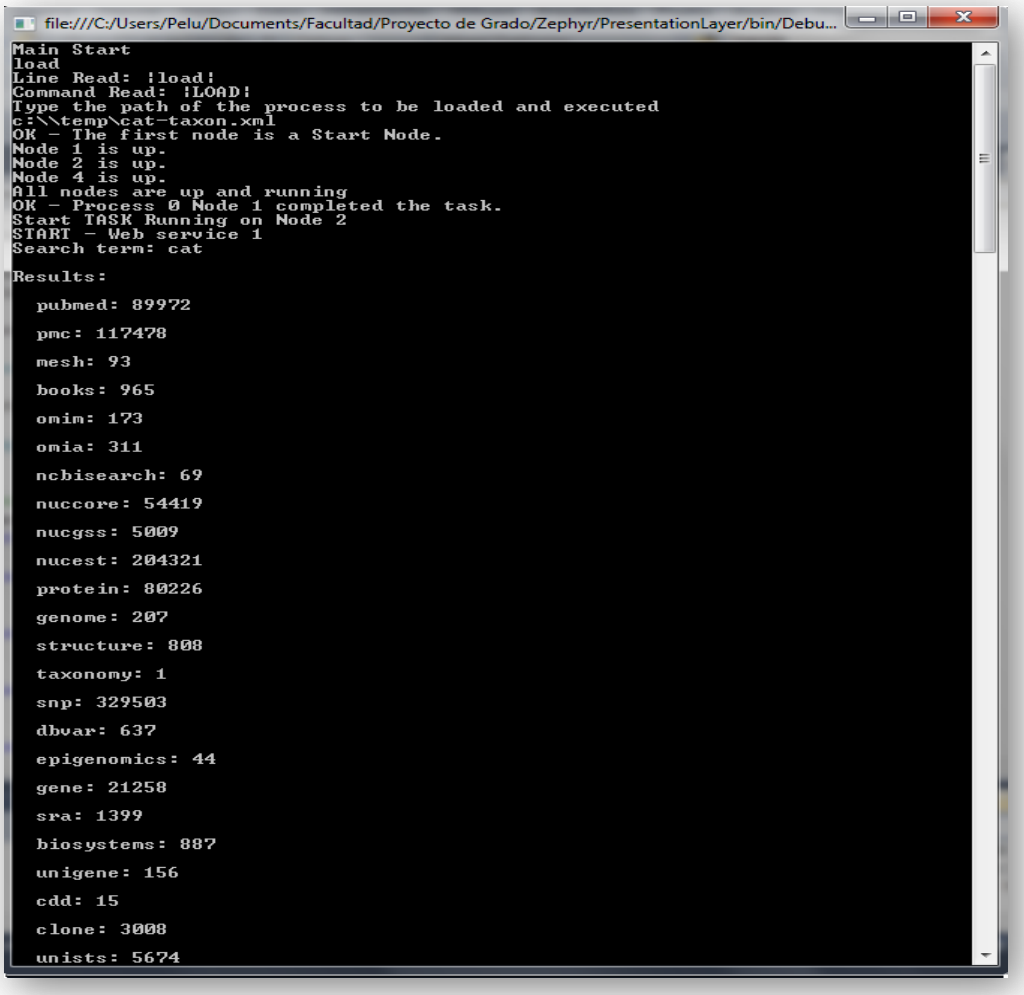

Imagen 22 - Consola de comandos

El usuario, desde la consola de comandos, también cuenta con la posibilidad de pausar un trabajo en ejecución, modificar el parámetro de un nodo de interés, para luego volver a ejecutar el flujo desde un punto deseado. Un ejemplo de resumen de proceso se representa en la Imagen23, en este caso cambiando el previo término de taxonomía "cat" por el término "mouse".

Los comandos aceptados vía consola se detallan en la sección 9.1.2 - Interfaz Consola.

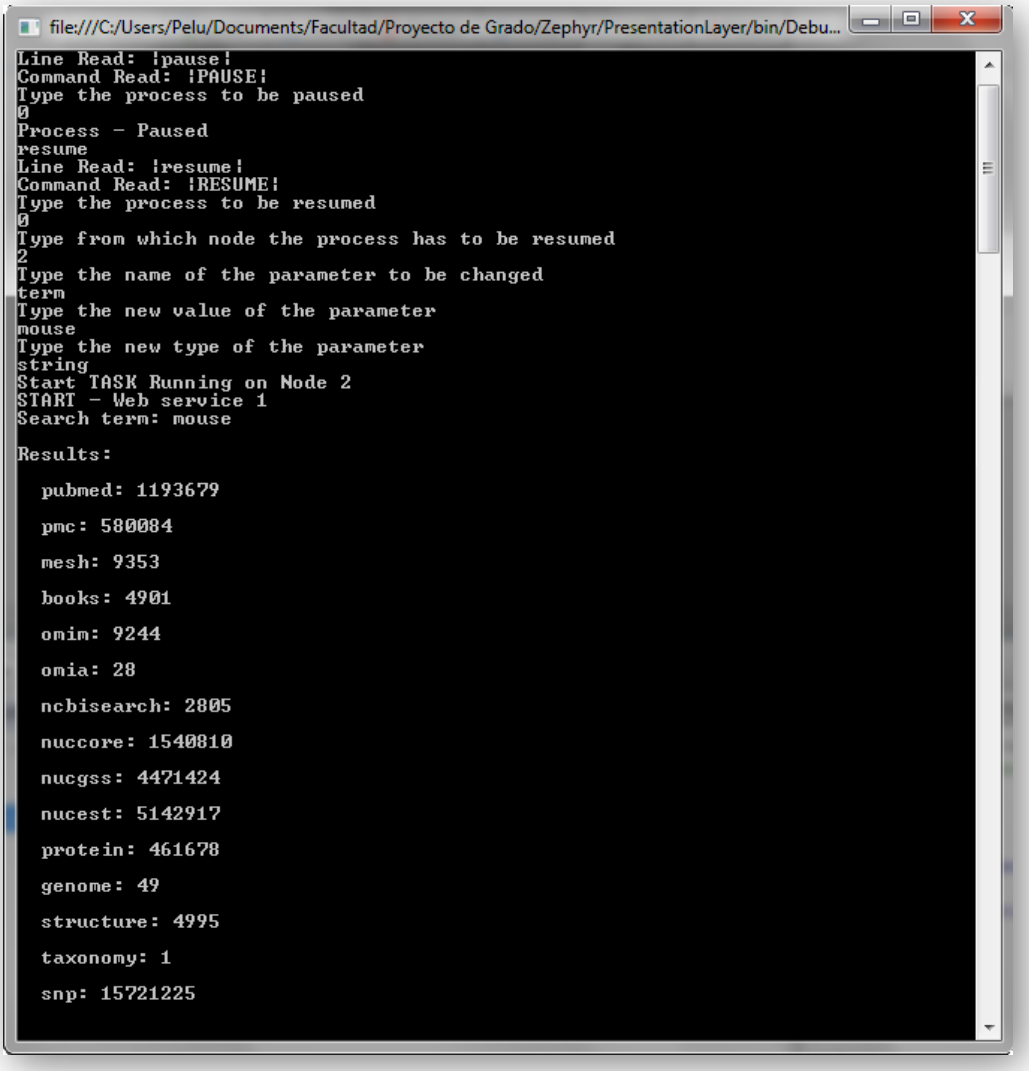

Imagen 23 - Pausa - Edita - Retoma

Interfaz Gráfica

Con la finalidad de facilitar el trabajo de crear y ejecutar flujos de trabajo, se creó una interfaz de usuario gráfica. La misma cuenta con componentes drag & drop, pretendiendo facilitar la tarea de compresión y utilización para el usuario.

Dentro de las prestaciones ofrecidas, se le da al usuario la posibilidad se cargar flujos previamente guardados.

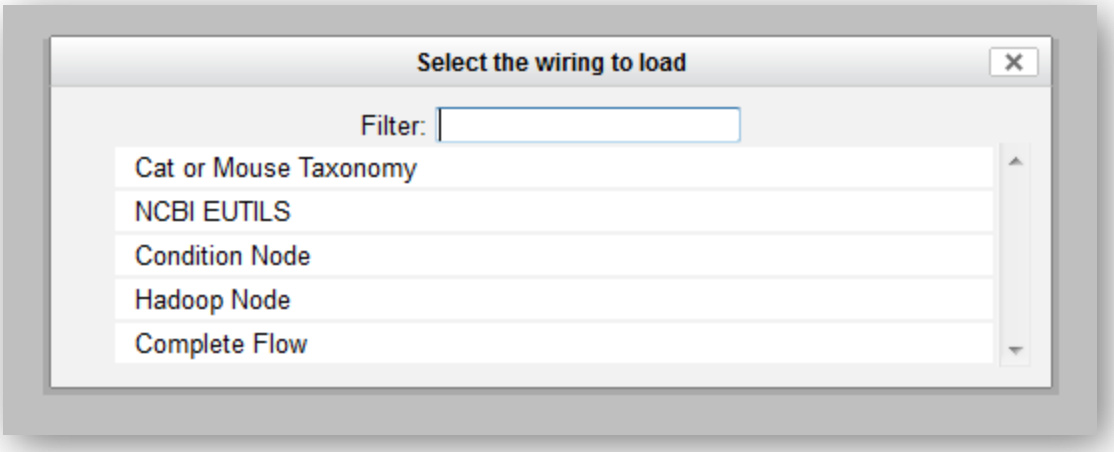

Imagen 24 - Trabajos Guardados

También ofrece un menú con los distintos nodos a utilizar, agregándose a los ya presentados en la sección 4.1.3 - Tipos de Nodo, nodos que representan inputs manuales o incluso comentarios.

Los enlaces representan una relación de precedencia entre nodos, son representados mediante flechas, mientras que los que representan entradas o parámetros se representan mediante arcos. Tal como se muestra en la Imagen25.

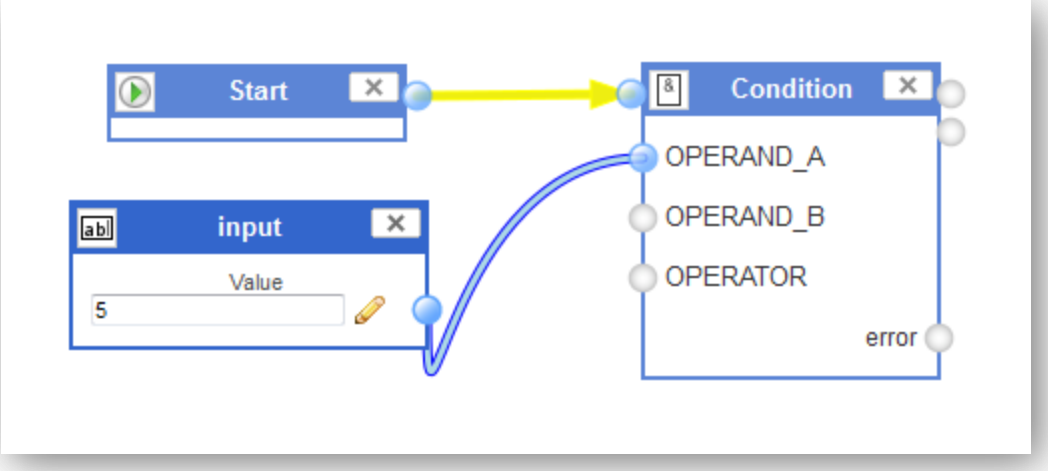

Imagen 25 - Nodos y Enlaces

Para proveer un mayor entendimiento del flujo, se presentan también un mini mapa que resume la estructura general y una sección para realizar anotaciones asociadas. Una representación general de la interfaz web queda representada en la Imagen26.

Para mayor información sobre la interfaz gráfica Web, consultar el manual de usuario en la sección 9.1.3 - Interfaz Gráfica.

Es importante notar que la herramienta ofrece la opción de almacenar estructuras de flujo de trabajo para su posterior reutilización o modificación, adición de notas y comentarios asociados al proyecto.

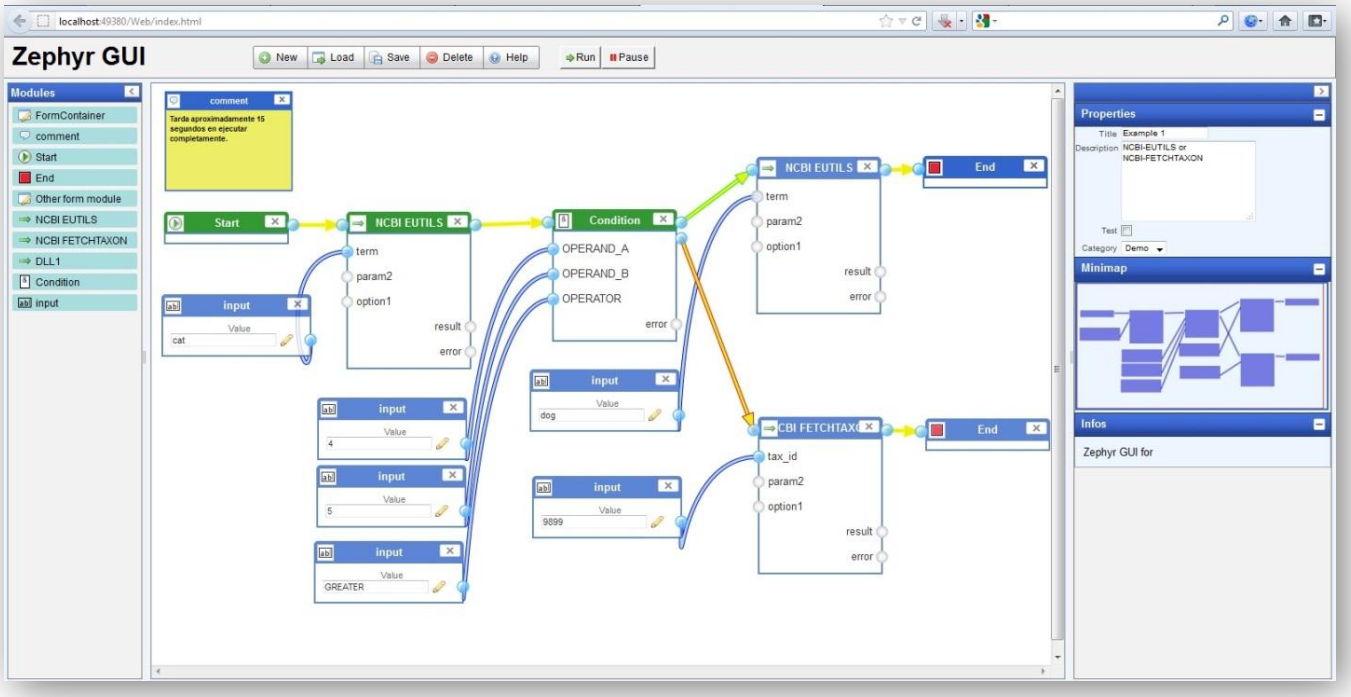

Imagen 26 - WEB GUI

4.5 Metadata

Esta sección detalla la metadata perteneciente a cada uno de los componentes y actividades involucradas en el flujo de datos. De esta manera se permite un mayor entendimiento del funcionamiento y en caso de ser necesario, facilita la extensión de la solución.

4.5.1 Nodos o Actividades XML

En la [Tabla4] se detalla la información general de cada nodo.

Nombre del Tag	Significado
processId	Identificador del proceso de ejecución al cual pertenece el nodo.
id	Identificador a nivel del proceso del nodo.
nodeType	Identificador de tipo de nodo. Indica que rol desempeña el nodo en el flujo. Los tipos de nodos son descriptos en la

	sección Tipos de Nodo
parameters	Sección que define la lista de parámetros seteados en dicho nodo.
maxIndex	Último identificador de mensaje recibido.
name	Nombre del parámetro seteado.
pamValue	Valor del parámetro seteado
type	Tipo del parámetro seteado (ya sea string, integer, boolean u otro).
action	Acción a realizar según se dese continuar el flujo, hacer streaming o editar un parámetro.

Tabla 4 - Metadata General Nodo

Cada nodo mantiene también una lista con la información de los nodos entrantes y salientes respectivamente, como se detalla en la [Tabla5].

Nombre del Tag	Significado
outgoingNodes	Lista de identificadores de nodos que tiene el nodo actual como salida.
incomingNodes	Lista de identificadores de nodos que tiene el nodo actual como entrada.
visited	Indicador para verificar si el nodo entrante culminó su ejecución.

Tabla 5 - Metadata Nodos Entrantes y Salientes

4.5.2 Nodos o Actividades JSON

Cada flujo de trabajo que involucra la interfaz web tiene asociada información general del proceso. Dicha información se detalla en la [Tabla6].

Nombre del Tag	Significado
name	Nombre que identifica el flujo de trabajo.
description	Descripción asociada al flujo de trabajo.
category	[TEST, DEMO, PRODUCTION]. Categorización del workflow.

language	Nombre del lenguaje para realizar subcategorización interna de divisiones.
working	Estructura contenedora para la información de nodos y asociaciones.

Tabla 6 – Metadata estructura general JSON

La metadata asociada a cada nodo se detalla en la [Tabla7].

Nombre del Tag	Significado
moduleId	Identificado de una nodo dentro del flujo.
name	Nombre asociado al tipo de nodo.
config	Sección que detalla la configuración de un nodo.
position	Coordenadas X, Y del nodo dentro del canvas de trabajo.
xtype	Tipo de nodo.
inputParams	Lista de parámetros introducidos.
inputValue	Valor del parámetro de entrada establecido.
inputType	Tipo del parámetro seteado (ya sea string, integer, boolean u otro).

Tabla 7 - Metadata Nodo JSON

Para las estructuras definidas en JSON, se mantiene una metadata específica para los links, detallada en la [Tabla8].

Nombre del Tag	Significado
wires	Sección que detalla la información de los links existentes.
src	Sección que detalla la información del origen de un link.
tgt	Sección que detalla la información sobre el desitino de un link.
moduleID	Identificador de un nodo, ya sea el origen o destino de un link.
terminal	Identifica con un nombre, cuál de los puntos de conexión ofrecidos por el nodo es el correspondiente al origen o destino del link.
drawingMethod	[straight, Bezier, Arrow] Forma visual de presentar los

	links.
color	Color del link
borderColor	Color del borde de un link
borderWidth	Ancho del borde de un link

Tabla 8 - Metadata Link JSON

4.6 Consideraciones

Entre las consideraciones de la solución encontramos las enumeradas en las siguientes secciones del presente apartado.

4.6.1 Creación de Colas

En forma previa a la ejecución se deben crear las colas de mensajes de Microsoft que implementarán la lógica de comunicación entre nodos, clientes y el servidor.

Por este motivo se mantiene una cola de mensajes para cada tipo de comunicación:

- Comunicación entre nodos del workflow.
- Comunicación entre el cliente consola y el servidor.
- Comunicación entre el cliente web y el servidor.

4.6.2 Creación de Nuevos Nodos

Ante la necesidad de interactuar con nuevas librerías o servicios web, se generó una lógica fácilmente extensible pero que requiere no obstante la previa carga de la definición del servicio así como la especificación de parámetros de entrada, salida y valores por defecto.

4.7 Pruebas de aceptación

Para las pruebas de aceptación se realizaron flujos de trabajo con servicios web y condicionales, los cuales fueron pausados, editados, retomados y ejecutados en streaming adicionalmente a su ejecución normal en una fase.

4.7.1 Ejecución de Servicios Web

Como parte de las pruebas de aceptación, se integraron servicios web pertenecientes al NCBI como nodos disponibles para la creación de estructuras de Workflow.

Servicio eUtils

El servicio eUtils permite la consulta de siete utilidades de NCBI (ESearch, ESummary, EFetch, ELink, EPost, EGQuery and ESpell). [Ref27] [Ref28]

- **ESearch**: Provee una lista de UID´s que machean con un término introducidos.
- **ESummary**: Retorna resúmenes de documentos en base a UID´s introducidos.
- **EFetch**: Retorna datos obtenidos del Entrez History Server a partir de una lista de UID´s.
- **ELink**: Retorna links y asociaciones a otros UID´s a partir de un UID introducido.
- **EPost**: Permite enviar una lista de UID´s al Entrez History Server.
- **EGQuery**: Retorna la cantidad de entradas encontradas en todas las bases de datos de Entrez, asociadas a un término en particular.
- **ESpell**: Retorna todas las formas de nombrar un término.

En particular, el ejemplo de ejecución mostrado anteriormente realiza un cruzamiento entre las distintas bases de datos bioinformáticas buscando el término “cat”, obteniendo en cada caso la cantidad de anotaciones asociadas al término en cada base de datos.

La función invocada es *run_eGquery()* la cual toma como parámetro el término buscado y opcionalmente una dirección de email donde serán enviados los resultados de la búsqueda.

Servicio efetch_taxon

El servicio web efetch_taxon es un expuesto por la NCBI que tiene como función la identificación taxonómica del organismo ingresado como parámetro. [Ref26] [Ref25]

Los parámetros de entrada aceptados por este servicio son los siguientes:

- **Term**
- **db**

En forma mandatoria se debe ingresar el parámetro de entrada term el cual se corresponde con el organismo a buscar y opcionalmente se puede ingresar la base de datos de internet en donde se desea realizar la búsqueda, la cual es por defecto pubmed.

4.7.2 Pausa, Edición y Retoma

Se realizaron pruebas pausando flujos de trabajo, editándolos y retomándolos. Bajo dichas pruebas primeramente se constató que la salida del flujo difiriera de acuerdo al parámetro de entrada ingresado al servicio.

En un primer lugar se ejecutó un flujo que invocaba al servicio web eUtils con un término determinado. Posteriormente se pausó y retomó el flujo cambiando dicho parámetro. Por último se verificó que la salida del mismo se viese modificada.

4.7.3 Streaming

Se efectuaron pruebas de streaming de datos ingresando en streaming el parámetro con el cual invocar al servicio efetch_taxon. Bajo dicho paradigma se mantuvo un nodo de streaming que envío en forma fraccionada el parámetro de entrada al nodo taskNode que invocaba al servicio efetch_taxon.

Dicho parámetro de entrada que se ingresaba en forma fraccionada ordenaba la búsqueda del término con el cual invocar al servicio de disco. De esta forma se simulaba el caso real de archivos del orden de varios terabytes que deben ser enviados en streaming y obtenidos de disco para no saturar las colas de mensajes.

4.7.4 Ejecución en Paralelo

Se cargaron 2 flujos de trabajo en el servidor y se ejecutaron en paralelo constatándose en esta prueba que el servidor permite la ejecución concurrente de procesos.

5. Trabajo futuro

5.1 Nuevos Tipos de Nodo

Actualmente el motor de workflow modela flujos con los siguientes tipos de nodo:

- Nodo de Inicio: StartNode
- Nodo de Fin: EndNode
- Nodo que evalúa condiciones: ConditionNode
- Nodo que ejecuta servicios web: TaskNode
- Nodo que realiza streaming de datos: StreamingNode

Sería deseable como trabajo a futuro el agregado de los siguientes nuevos tipos de nodo:

- Nodo De Iteración: nodo For que permita la ejecución de cierta porción del flujo con distintos datos obtenidos en streaming por el StreamingNode
- Nodo Fork: nodo que permita separar en dos flujos al proceso que se encuentra ejecutando
- Nodo Join: nodo que permita unir los flujos separados por el nodo fork
- Nodo Email: actualmente la interfaz gráfica ofrece soporte para la creación de nodos que envían emails. Es recomendable el agregado de este nodo para la notificación al científico de cuando un flujo finalizó.

5.2 Persistencia en Base de Datos

Actualmente el flujo se modela en xmls pudiéndose cargar y salvar en este formato. Sería sin embargo recomendable, agregar a esta forma de modelado, el almacenamiento de procesos en base de datos, para que de esta forma, el científico pueda si lo desea monitorear por base el flujo, editar parámetros o ver el resultado de viejas ejecuciones.

No obstante que esta información no se almacena en base de datos, la misma en registrada en logs y xmls que permiten la ejecución de estas tareas por parte del usuario.

5.3 Sistema de Filtros

Las colas de mensaje que se utilizan actualmente son compartidas para todos los flujos que se encuentran ejecutando. Es decir, si dos flujos se encuentran ejecutando en forma simultánea, los mismos utilizarán las mismas colas para comunicarse. Por este motivo, los mensajes deben encontrarse dirigidos a procesos y nodos en particular, y así son enviados por la cola a todos los nodos, los cuales consultan el mensaje y en caso de no pertenecerles, lo devuelven a la cola para ser leídos por otro nodo.

Sería recomendable realizar el envío de mensajes aprovechando el sistema de filtros de las colas de Windows, este sistema permitiría reducir el tiempo en que el nodo al cual se le desea enviar un mensaje recibe el mismo. El sistema de filtros, a través del uso de etiquetas a nivel de nodos, permite que los mismos, solo procesen los mensajes dirigidos a ellos mismos.

5.4 Nodo Map Reduced

Debido a la arquitectura distribuida de la solución, especialmente, la posibilidad de separar la instancia de los nodos del servidor Main, se puede implementar un nuevo tipo de nodo que herede de la clase Node y que permita la implementación de la técnica Map Reduced. De esta forma, se podría aprovechar la capacidad del entorno cloud para ejecutar algoritmos complejos.

5.5 Interfaz Gráfica

La interfaz gráfica actualmente permite el modelado, la ejecución y el monitoreo de procesos. No obstante, no permite la ejecución de la funcionalidad pausa y retoma. Sería recomendable el agregado de esta funcionalidad. El impacto de agregar la misma es únicamente a nivel de la interfaz gráfica dado que el motor de workflow ya soporta dicha funcionalidad.

5.6 Generalización de Invocación a Librerías

Actualmente, la invocación a nuevas librerías requiere un mínimo de programación. El proceso conlleva mínimo trabajo, únicamente se debe crear un nuevo nodo que extienda al nodo TaskNode y que contenga la url del servicio web a invocar, pero requiere que el usuario posea conocimientos informáticos.

Sería recomendable ajustar el nodo TaskNode para que el agregado de librerías sea automático y no requiera programación.

5.7 Entorno Clúster & Cloud

El motor del workflow construido permite la ejecución en clúster y en entorno PC standalone, no así, la ejecución en un entorno cloud.

Sería recomendable extender el motor, para permitir la ejecución en el entorno de Cloud Computing: Microsoft Azure.

5.8 Microsoft Biology Foundation

Como se explicó en la sección 2.3.6 - Microsoft Biology Foundation, el framework Microsoft Biology Foundation ofrece una serie de herramientas al científico para la manipulación y el procesamiento de información bioinformática. Dicho framework se encuentra como una extensión al framework .NET de Microsoft.

Como trabajo a futuro se podría probar la integración del motor de workflow con estas herramientas a modo de extender las funcionalidades bioinformáticas que el mismo provee.

6. Conclusiones

Luego de haber realizado trabajos de investigación e implementación en el área de la bioinformática, se pudo obtener un conjunto de conclusiones con carácter diferente en cada caso.

Respecto a la documentación, se observó que los distintos servicios y herramientas ofrecidas, cuentan con una documentación vaga, escasa o en algunos casos, nula. Esto claramente dificulta la tarea de los investigadores u otros usuarios al momento de requerir apoyo para su investigación.

Por otro lado, el volumen de datos hoy en día producido, es de tamaño inabarcable para los métodos y poder de cómputo actualmente disponibles, tales como Desktop o Clúster. Las bases de datos bioinformáticas, inevitablemente mantienen datos muchas veces redundantes o repetidos. La utilización del Cloud podría solventar el problema del poder de cómputo disponible.

Los sistemas de workflow existentes marcan una tendencia a la unificación de trabajo en toda la comunidad científica, proveyendo entornos compatibles y sitios para que los usuarios compartan sus trabajos. Aun así, no existe una estandarización de patrones de diseño de workflows científicos, lo cual implica la existencia de implementaciones propias de cada fabricante.

En cuanto a los métodos de trabajo, se hace notorio que muchos bioinformáticos realizan trabajos de una forma aún muy artesanal, corriendo pequeños scripts muchas veces de implementación propia para resolver problemas puntuales. En esta área es donde se hace más notoria la necesidad de la cooperación e integración con personas del sector informático, pudiendo así optimizar y reducir los costos y tiempos de cómputo.

La idea de pausar, editar y reanudar un trabajo y permitir el streaming de datos entre dos nodos resultan conceptos innovadores y de gran utilidad para el trabajo de la comunidad bioinformática. Hasta el momento, no se conocen trabajos que desarrollen estas funcionalidades sobre workflows científicos.

También, el hecho de utilizar colas de mensajes para la comunicación entre componentes permite una gran escalabilidad y la adaptación a entornos Clúster o Cloud. Las herramientas existentes al momento de implementar el motor de workflow resultaron ser rígidas en cuanto a su estructura, por lo cual no se facilita su movilidad

entre entornos. Para este caso, tampoco se conocen trabajos que estén basados en este tipo de arquitecturas.

Por último, la realización del proyecto de grado sobre la temática bioinformática, nos generó un profundo interés en el tema que nos condujo a querer ahondar en el mismo. Actualmente nos encontramos prosiguiendo nuestros estudios formales en la temática y relacionándonos con gente del área.

La bioinformática es un campo con un gran potencial de crecimiento que todavía tiene mucho camino por recorrer, siendo así el trabajo interdisciplinario la puerta a la apertura de esta fuente de información que encierra el conocimiento profundo del organismo vivo.

7. Glosario

7.1 Informático

1. **GPU –** GPU Computing es el uso de la GPU (unidad de procesamiento gráfico) junto con una CPU para acelerar operaciones de cálculo científico o técnico de propósito general.

2. **Clúster –** Conjunto de máquinas que se comportan como si fuesen una única computadora.

3. **Cloud Computing –** Es un modelo que permite el acceso según conveniencia a un conjunto de recursos informáticos sin importar ubicación física o configuración de los mismos. Se basa en el alojamiento del software o utilización de recursos en data centers privados accesibles mediante Internet.

4. **GUI -** Graphical User Interface

5. **IIS –** Internet Information Services

6. **WSBPEL –** Web Services Business Process Execution Language

7. **Windows Azure –** Plataforma Microsoft de Cloud Computing

8. **XML -** Extensible Markup Language permite definir la gramática de lenguajes específicos para estructurar documentos.

9. **JSON -** JavaScript Object Notation es un formato ligero para intercambio de datos

7.2 Biológico

1. **Gen** - Un gen es la unidad física y funcional de la herencia, que se pasa de padres a hijos. Los genes están compuestos por ADN y la mayoría de ellos contiene la información para elaborar una proteína específica. Cada gen tiene una localización específica en un determinado cromosoma, y el conjunto de todos los genes, contenidos en todos los cromosomas, constituye el genoma.

2. **Biochip –** Los Biochips, también conocidos como Microarrays son una superficie sólida a la cual se une una colección de fragmentos de ADN. Estos chips de ADN se usan para analizar la expresión diferencial de genes, monitorizándose los niveles de miles de ellos de forma simultánea.

3. **Cromosoma** - Los cromosomas están constituidos por ADN (ácido desoxirribonucleico), que codifica la información hereditaria, y por proteínas histónicas y no histónicas. Cada cromosoma está formado por una única molécula de ADN, en la que cada gen ocupa un segmento.

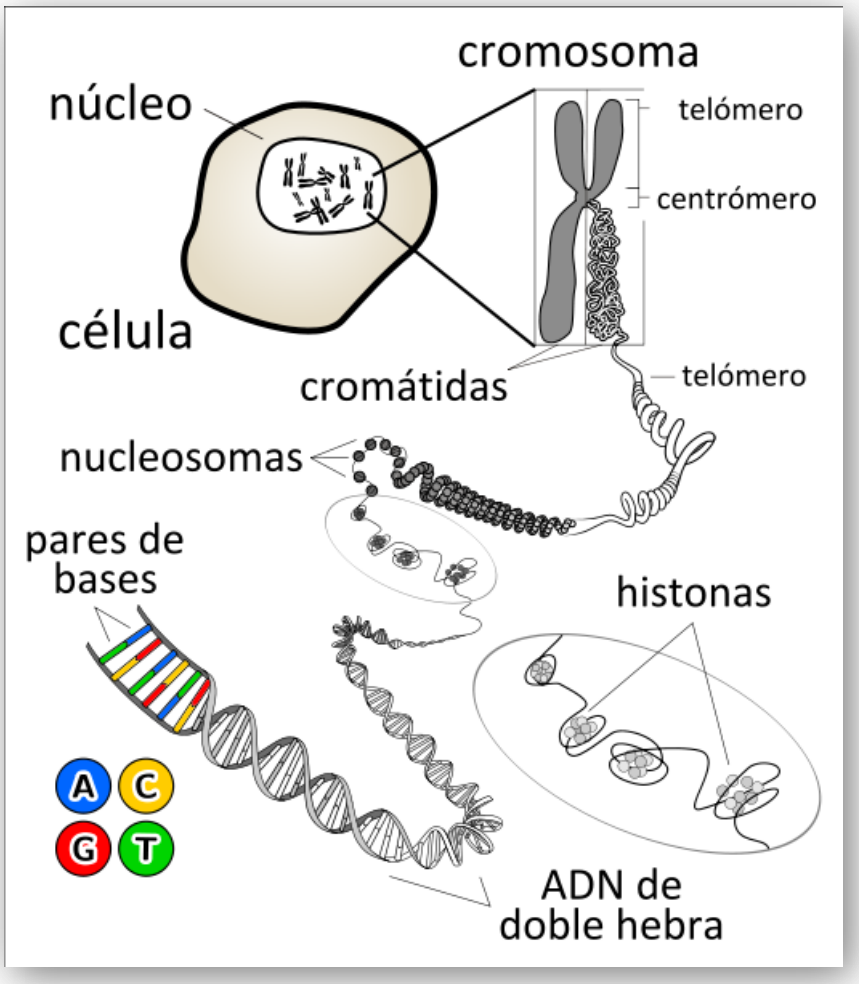

Imagen 27 – Transcripción y Traducción

4. **ADN -** El ADN está constituido por la asociación de moléculas llamadas nucleótidos, formadas por la unión de una molécula de fosfato, una del azúcar desoxirribosa y una base nitrogenada. Ya que cuatro bases distintas, adenina, guanina, timina y citosina participan en la formación de los nucleótidos, hay cuatro tipos distintos de estos. Para formar ADN, los nucleótidos se vinculan por sus grupos fosfato y conforman una larga hebra, cuyas bases nitrogenadas se unen por uniones débiles pero muy específicas con las de otra hebra. Se forman así pares de bases, que determinan que ambas hebras, apareadas, se enrollen para dar lugar a la estructura de doble hélice. Las uniones entre las bases solo ocurren, por una parte, entre la adenina y la timina y, por otra, entre la guanina y la citosina, las que por eso se llaman bases complementarias. La especificidad de las uniones entre bases determina la conservación y la transmisión de la información hereditaria.

   El mensaje de la herencia o código genético está contenido en el orden o secuencia con que las bases aparecen en la larga hebra del ADN.

   El mensaje genético solo consiste en información que determina el número, el tipo y la secuencia de aminoácidos de cada uno de los distintos tipos de proteínas de un organismo, La secuencia de bases del ADN determina la secuencia en que los aminoácidos se enlazan entre sí para dar lugar a una proteína.[Ref29]

5. **Genoma Humano -** El genoma humano es la secuencia de ADN contenida en 23 pares de cromosomas en el núcleo de cada célula humana diploide.

6. **Expresión Génica -** La expresión génica es el proceso por medio del cual todos los organismos procariotas y eucariotas transforman la información codificada en los ácidos nucleicos en las proteínas necesarias para su desarrollo y funcionamiento. En todos los organismos, inclusive los eucariotas, el contenido del ADN de todas sus células es idéntico. Esto quiere decir que contienen toda la información necesaria para la síntesis de todas las proteínas. Pero no todos los genes se expresan al mismo tiempo ni en todas las células. Hay sólo un grupo de genes que se expresan en todas las células del organismo y codifican proteínas que son esenciales para el funcionamiento general de las células y son conocidos como *"housekeeping genes".* El resto de los genes se expresan o no en los diferentes tipos de células, dependiendo de la función de la célula en un tejido particular.

7. **Filogenética -** La filogenética o filogenia es la parte de la biología que estudia la evolución de las especies de forma global.

8. **Contig o Cóntigo -** Colección de clones representativos del genoma que se solapan entre sí.

9. **Estructuras macromoleculares -** A menudo el término macromolécula se refiere a las moléculas que pesan más de 10.000 dalton de masa atómica. Son moléculas muy grandes y pueden ser tanto orgánicas como inorgánicas. [Ref30]

10. Secuencia genómica **-** Secuencias de ADN que caracterizan a un individuo.

11. **Expresión de genes -** Es el proceso por medio del cual todos los organismos transforman la información codificada en los ácidos nucleicos en las proteínas.

12. **Vía metabólica -** Es una secuencia ordenada de reacciones en las que el producto final de una reacción es el sustrato inicial de la siguiente.

13. **Árbol Taxonómico -** Es una estructura que permite agrupar a los seres vivos de forma jerárquica, según su parentesco y sus relaciones filogenéticas.

14. **Árbol Filogenético -** Es un árbol que muestra las relaciones evolutivas entre varias especies u otras entidades que se cree que tienen una ascendencia común.

15. **Primer -** Es una cadena de ácidos nucleicos o de una molécula relacionada que sirve como punto de partida para la replicación del ADN.

16. **PCR -** (Polymerase Chain Reaction) es una técnica cuyo objetivo es obtener un gran número de copias de un fragmento de ADN particular, partiendo de una cantidad muy pequeña. De esta manera resulta mucho más fácil identificar con una muy alta probabilidad los objetos buscados.

17. **Motif -** Es una secuencia breve de aminoácidos que suele servir de sitio de reconocimiento o asociada a una determinada función, contigua o alineada con respecto a ciertas posiciones invariables.

7.3 Algoritmia

1. **Maximal Unique Match -** se llama coincidencia máxima y única, cuando se da una coincidencia única entre dos secciones de dos secuencias tal que no existe una sección mayor que comprenda a estas y que permita una coincidencia.

2. **SAM –** *Sequence Alignment Map*, formato de archivo que permite representar en forma compacta e indexada, alineaciones de secuencias de nucleótidos.

3. **BAM –** *Binary Alignment Map* compresión binaria de un archivo en formato SAM.

8. Referencias

1. Definiciones Bioinformática
   Fecha Visitado:15/05/2012
   http://www.ebi.ac.uk/2can/bioinformatics/bioinf_what_1.html

2. Definiciones Bioinformática
   Fecha Visitado: 15/05/2012
   http://www.ncbi.nlm.nih.gov/pubmed/11552348

3. Definiciones Bioinformática
   Fecha Visitado: 15/05/2012
   http://lema.rae.es/drae/?val=bioinform%C3%A1tica

4. Definiciones Bioinformática
   Fecha Visitado: 16/05/2012
   http://www.pasteur.fr/recherche/unites/Binfs/definition/bioinformatics_definition.html

5. Workflows Científicos vs. Empresariales
   Fecha Publicación: Marzo 2009
   Nombre Publicación: Business versus Scientific Workflow: A Comparative Study
   Autores: Ustun Yildiz — Adnene Guabtni — Anne H.H. Ngu
   http://www.cs.ucdavis.edu/research/tech-reports/2009/CSE-2009-3.pdf

6. Herramientas Bioinformáticas - Taverna
   Fecha Publicación: 2010
   Nombre Publicación: Taverna: un ambiente para el desarrollo de experimentos científicos Autores: Guzmán Llambías y Raul Ruggia
   http://www.fing.edu.uy/inco/pedeciba/bibliote/reptec/TR1011.pdf

7. Herramientas Bioinformáticas - Trident
   Fecha Visitado: 19/06/2011
   http://research.microsoft.com/en-us/collaboration/tools/trident_workbench.doc

8. Herramientas Bioinformáticas - Taverna

Fecha Visitado 21/06/2011
http://www.taverna.org.uk/pages/wp-content/uploads/2010/04/T2Architecture.pdf

9. Workflow – Especificaciones YAWL
Fecha Visitado: 20/09/2011
http://sourceforge.net/projects/yawl/

10. Workflows - Patrones
Fecha Visitado: 20/09/2011
http://www.workflowpatterns.com/

11. Workflow – Especificaciones YAWL
Fecha Visitado: 15/09/2011
http://www.yawlfoundation.org/manuals/YAWLUserManual2.2.pdf

12. Workflow – Especificaciones WS-BPEL
Fecha Visitado 03/09/2011
http://docs.oasis-open.org/wsbpel/2.0/Primer/wsbpel-v2.0-Primer.pdf

13. Workflow – Especificaciones WS-BPEL
Fecha Visitado: 03/09/2011
http://bpel.xml.org/wiki

14. Definiciones Bioinformática
Fecha Visitado: 02/06/2012
http://www.ebi.ac.uk/2can/bioinformatics

15. Definiciones Bioinformática
Fecha Visitado: 10/06/2012
http://www.ncbi.nlm.nih.gov/

16. Bases de Datos Biológicas
Fecha Visitado: 21/06/2012
http://www.ncbi.nlm.nih.gov/sites/gquery

17. Bases de Datos Biológicas
Fecha Visitado: 21/06/2012

http://www.ebi.ac.uk/embl/Documentation/help/srshelp.html

18. Interfaz Gráfica

Fecha Visitado: 21/06/2012

http://neyric.github.com/wireit/

19. Herramientas Bioinformáticas

Fecha Visitado: 12/07/2012

http://research.microsoft.com/en-us/projects/bio/mbf.aspx

20. Herramientas Bioinformáticas

Fecha Visitado: 10/07/2012

http://mummer.sourceforge.net/examples/

21. Herramientas Bioinformáticas

Fecha Visitado: 10/07/2012

http://src.gnu-darwin.org/ports/biology/mummer/work/MUMmer3.20/nucmer

22. Herramientas Bioinformáticas

Fecha Visitado: 10/07/2012

http://mummer.sourceforge.net/examples/#nucmer

23. Herramientas Bioinformáticas

Fecha Visitado: 10/07/2012

http://samtools.sourceforge.net/samtools.shtml

24. Hadoop

Fecha Visitado: 12/07/2012

http://hadoop.apache.org/

25. WebService NCBI

Fecha Visitado: 14/07/2012

http://eutils.ncbi.nlm.nih.gov/soap/v2.0/efetch_taxon.wsdl

26. WebService NCBI

Fecha Visitado: 12/07/2012

http://www.ncbi.nlm.nih.gov/books/NBK25499/#chapter4.EFetch

27. WebService NCBI

Fecha Visitado: 12/07/2012

http://www.ncbi.nlm.nih.gov/soap/v2.0/eutils.wsdl

28. WebService NCBI

Fecha Visitado: 12/07/2012

http://www.ncbi.nlm.nih.gov/books/NBK43082/

29. Conceptos Bioinformática, Genética y Bioquímica

Fecha Visitado: 01/06/2012

http://www.cienciahoy.org.ar/ln/hoy67/conceptosgenetica.htm

30. Conceptos Bioinformática, Genética y Bioquímica
Libro: Principios de Bioquímica. Edición: 5ta. Autor: Lehninger

31. Workflows Científicos
Fecha de Visitado: 25/06/2012
Scientific Workflows: Scientific Computing Meets Transactional Workflows
Munindar P. Singh and Mladen A. Vouk
http://www.csc.ncsu.edu/faculty/mpsingh/papers/databases/workflows/sciworkflows.html

32. Workflows Científicos

Fecha de Visitado: 25/06/2012
Provenance Collection Support in the Kepler Scientific Workflow System
Ilkay Altintas, Oscar Barney, Efrat Jaeger-Frank
http://users.sdsc.edu/~altintas/KeplerProvenanceRecorder.pdf

33. Cloud Computing
Fecha Visitado 15/08/2012
http://opencloudmanifesto.org/Cloud_Computing_Use_Cases_Whitepaper-4_0.pdf

34. Complejidad Algorítmica
An Introduction to Bioinformatics Algorithms - Capítulo 4
Neil C. Jones,Pavel A. Pevzner

35. Complejidad Algorítmica
Bioinformatics Algorithms: Techniques and Applications - Capítulo 5.3
Ion Mandoiu,Alexander Zelikovsky

36. PEDECIBA
Fecha Visitado 15/05/2012
http://www.pedeciba.edu.uy

37. Proyecto Genoma Humano
Fecha Visitado 15/08/2012
All About The Human Genome Project (HGP)
http://www.genome.gov/10001772

38. A Workflow Approach to Stream Processing
Autor: ETH ZURICH Fecha Publicación: 2008

39. Streamflow - Programming Model for Data Streaming in Scientific Workflows
Autores: Chathura Herath/Beth Plale - School of Informatics, Indiana University, Bloomington, Indiana

9. Anexos

9.1 Manuales de Usuario

9.1.1 Configuración del Servidor

El ambiente se encuentra preparado para correr sobre el sistema operativo Windows 7.

En primer lugar, para la instalación del servidor en una PC standalone es necesario instalar las colas de mensajes de MSMQ desde la funcionalidad “turn windows features on or off” del sistema operativo Windows.

Una vez instaladas las mismas, se deben crear tres colas de mensajes privadas con los siguientes nombres:

- myWSQueue
- myMainQueue
- myQueue

Realizados los anteriores pasos, el servidor ser encuentra listo para ejecutar.

9.1.2 Interfaz Consola

La interfaz consola ofrece el siguiente listado de comandos:

Comando LOAD

El comando load tiene como función la carga y ejecución de un nuevo proceso en el entorno de ejecución. A continuación de la ejecución del comando load se despliega el siguiente comando en pantalla:

```
Type the path of the process to be loaded and executed
```

Luego del despliegue del anterior mensaje se debe ingresar la ruta en donde se encuentra e XML que modela al mismo.

Los resultados de la ejecución se registrarán en pantalla.

Comando PAUSE

El presente comando permite el pausado del flujo de carga. Se espera que el usuario ejecute el mismo cuando constate que alguno de los parámetros de entrada que ingresó no le retornó los resultados esperados. En ese caso, el usuario pausa el flujo y procede a continuación a retomarlo editando un parámetro.

El usuario luego de digitar el comando pause, visualiza el siguiente mensaje en pantalla:

```
Type the process to be paused
```

Ante dicho mensaje el usuario debe ingresar el identificador del proceso que desea pausar.

Comando RESUME

El comando resume permite retomar un flujo editando un parámetro de entrada. El usuario debe ingresar el comando y a continuación visualizar el siguiente mensaje:

```
Type the process to be resumed
```

En respuesta al mensaje anterior el usuario debe ingresar el identificador del proceso a ser retomado.

```
Type from which node the process has to be resumed
```

Luego del anterior mensaje, el usuario debe ingresar desde cual nodo desea que el proceso retome la ejecución. Los nodos anteriores al mismo no volverán a ejecutar, y los resultados almacenados en los mismos permanecerán incambiados.

```
Type the name of the parameter to be changed
```

El anterior mensaje consulta cual es el parámetro de entrada que se debe cambiar para el nodo desde donde se desea retomar. En este punto se debe ingresar el nombre del parámetro tal cual figura en el XML que modela el flujo.

```
Type the new value of the parameter
```

Luego de esta consulta se debe ingresar el nuevo valor del parámetro anteriormente especificado.

```
Type the new type of the parameter
```

Por último, luego de este mensaje se debe ingresar el tipo del parámetro, pudiendo ser el mismo un string, un boolean, etc..

EXIT

El comando exit tiene como finalidad detener la ejecución del servidor y de todos los procesos que se encuentran ejecutando en el momento.

9.1.3 Interfaz Gráfica

Requisitos

Para poder ejecutar la interfaz de usuario web, es necesario cumplir con los siguientes requisitos:

- Tener un ambiente con interfaz de consola previamente configurado y funcionando correctamente (punto anterior).

- Instalación del servidor Internet Information Services desde la funcionalidad “turn windows features on or off” del sistema operativo Windows.

- Contar con un directorio nombrado “temp” con permisos de lectura y escritura en la raíz de la unidad de disco C.

- Utilizar un browser de internet que tenga habilitado JavaScript y en versiones:
    - Firefox (2+)
    - Internet Explorer (6+)
    - Safari (3+)
    - Opera (10.6+).

Utilización

Para acceder a la interfaz web, es necesario ingresar en la URL del browser a utilizar la ruta referente al sitio:

http://[nombre del servidor]/[nombre de carpeta en el servidor]/web/index.html

Si la configuración es correcta se desplegará una página web titulada “Zephyr GUI”, dependiendo de configuración de visualización del usuario, similar a la presentada en la Imagen 28.

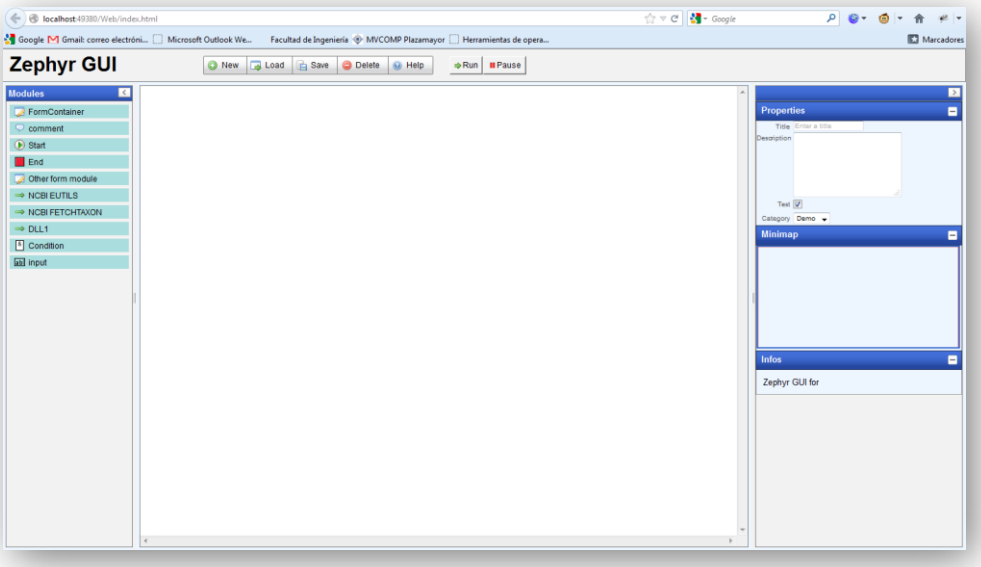

Imagen 28 - Interfaz de Usuario Web

La herramienta Web se puede descomponer en las siguientes secciones:

- **1- Herramientas de Control**: Las herramientas de control permites realizar acciones sobre un workflow. Dentro de estas acciones se permite:

    - Creación de un nuevo Workflow
    - Carga de un Workflow existente
    - Salvar una estructura creada
    - Borrar una estructura creada
    - Obtener ayuda de funcionamiento
    - Ejecutar una estructura de Workflow creada
    - Pausar un Workflow en funcionamiento

- **2- Módulos disponibles**: En esta sección se presentan los tipos nodos y módulos disponibles.

- **3- Propiedades del Workflow**: Dentro de las propiedades del workflow se presenta el nombre otorgado al mismo (obligatorio), una descripción del mismo y la posibilidad de establecer el tipo de ejecución que se está realizando.

- **4- Canvas o lienzo**: En el lienzo de trabajo, se controla el flujo de trabajo, la precedencia entre nodos las relaciones de comunicación entre ellos. Los módulos y links se posicionan en el lienzo con acciones Drag & Drop desde la sección de Módulos.

- **5- Minimapa**: El minimapa es una representación miniaturizada del lienzo de trabajo. Éste permite tener una visualización resumida de la interacción entre nodos en caso de que la estructura de workflow sea muy extensa.

Las distintas secciones de la interfaz de usuario Web se detallan en la Imagen29

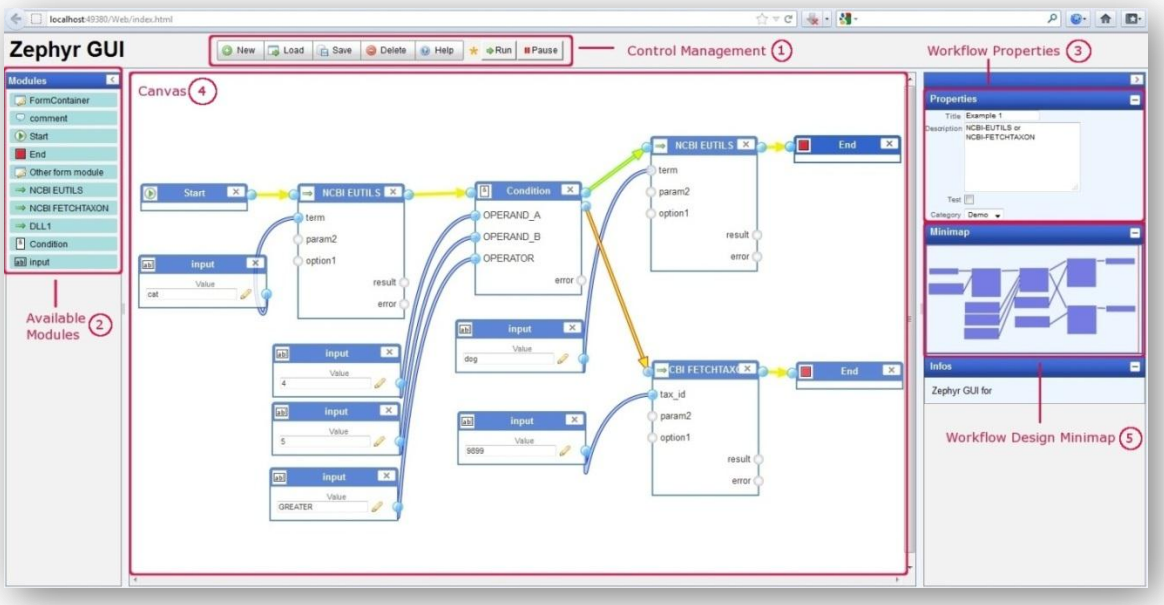

Imagen 29 - Secciones Interfaz Web

Carga de Estructuras

La herramienta de interfaz gráfica presenta la posibilidad de cargar y guardar estructuras de workflow. Al seleccionar la opción de "Load" en el panel de "Herramientas de Control", un menú (Imagen30) se desplegará mostrando las estructuras previamente almacenadas.

También se provee un filtro para realizar búsquedas sobre los nombres de las estructuras almacenadas.

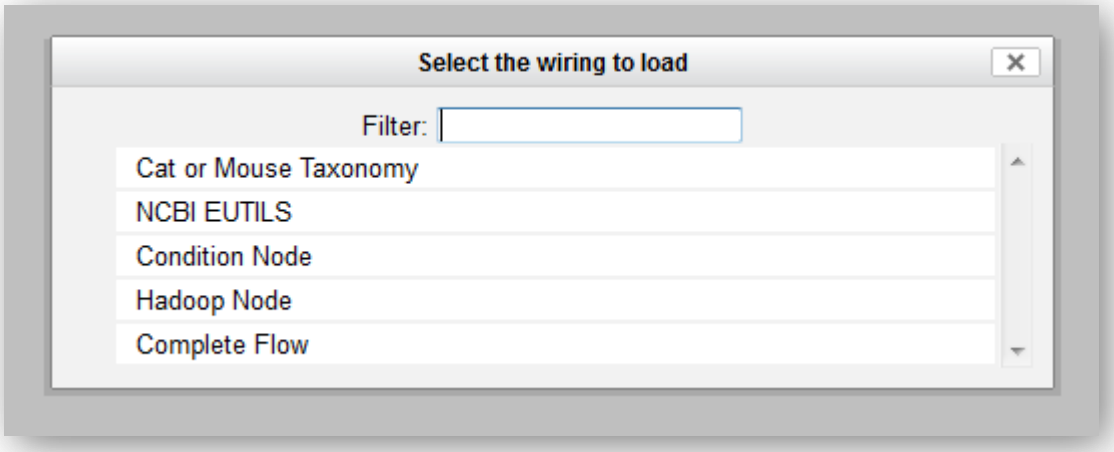

Imagen 30 - Panel de Carga de Estructuras

Guardar Estructuras

Una estructura puede ser almacenada para su posterior utilización. Esta funcionalidad se encuentra disponible en el panel de "Herramientas de Control". Cabe aclarar que una estructura debe contar con un título introducido en el "Panel de Propiedades" para poder ser almacenada.

Módulos

Un módulo puede representar un servicio a invocar, un parámetro de entrada/salida (ya sea un valor ingresado a mano o la ruta de un archivo), un nodo de incio/fin o un nodo que representa notas en el flujo.

Los módulos cuentan (en caso de ser nodos del Workflow) con uno o más conectores de entada o salida para establecer precedencia entre nodos o establecer el orden del flujo. Opcionalmente cuentan con conectores para establecer parámetros de entrada/salida. No está permitido asociar conectores que definen flujo con conectores que definen parámetros.

En caso de tratarse de módulos de parámetros de entrada, éstos proporcionan la posibilidad de setear propiedades del parámetro, como el tipo, largo o si se trata de un parámetro de carácter obligatorio, entre otros. Un ejemplo se detalla en la imagen31.

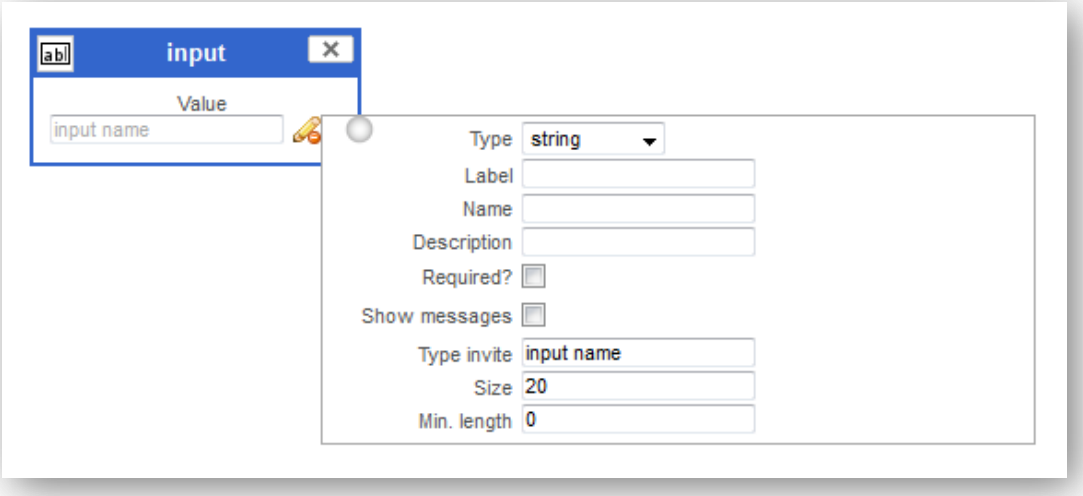

Imagen 31 - Módulo de tipo parámetro

Links

La asociación entre nodos, tal como se detalla en la Imagen32, se diferencia por el tipo de conector. Los links de precedencia se representan con flechas rectas, mientras que los parámetros de entrada se representan con arcos.

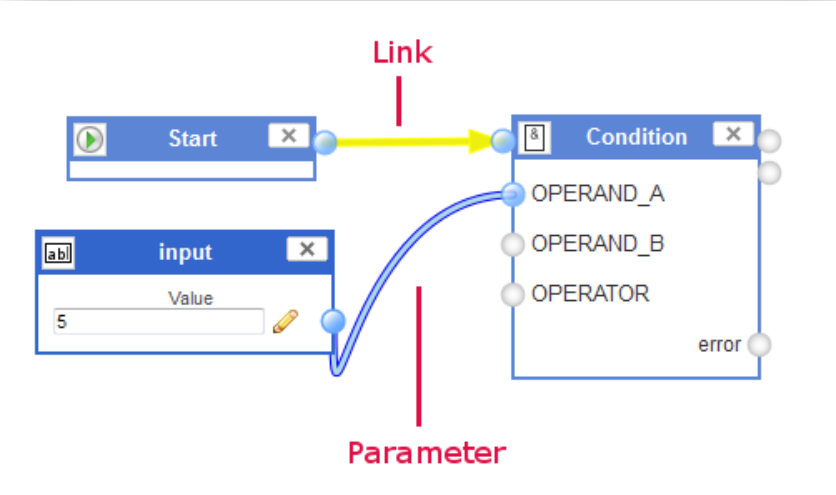

Imagen 32 - Asociación entre nodos

Outputs

Los outputs de los nodos se presentarán en los archivos establecidos como parámetros de salida o en caso contrario, en la consola de usuario.

9.2 Estructuras

Respuesta de estado de ejecución asociado a Imagen 26 - WEB GUI y modelo de Imagen 19 - Modelo de Flujo.

```
[{"Id":14,"visited":true},
{"Id":13,"visited":true},
{"Id":10,"visited":true},
{"Id":9,"visited":true},
{"Id":6,"visited":true},
{"Id":4,"visited":false},
{"Id":2,"visited":true},
{"Id":1,"visited":false}]
```

Estructura XML representando modelo de Imagen 26 - WEB GUI y modelo de Imagen 19 - Modelo de Flujo.

```
<?xml version="1.0" encoding="utf-8"?>
<Structure xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="http://www.zephyr.com">
  <_nodes>
    <NodeTO>
      <outgoingNodes>
        <int>2</int>
      </outgoingNodes>
      <parameters />
      <processId>0</processId>
      <id>1</id>
      <nodeType>START_NODE</nodeType>
      <incomingNodes />
    </NodeTO>
    <NodeTO>
      <outgoingNodes>
        <int>4</int>
      </outgoingNodes>
      <parameters>
        <Parameter>
          <maxIndex>0</maxIndex>
          <name>WS_TYPE</name>
          <pamValue>WEBSERVICE_1</pamValue>
          <type />
          <action>NEW</action>
        </Parameter>
        <Parameter>
          <maxIndex>0</maxIndex>
          <name>term</name>
          <pamValue>cat</pamValue>
          <type>string</type>
          <action>NEW</action>
        </Parameter>
      </parameters>
      <processId>0</processId>
      <id>2</id>
```

```
      <nodeType>TASK_NODE</nodeType>
      <incomingNodes>
        <DataTypeVisitId>
          <Id>1</Id>
          <visited>false</visited>
        </DataTypeVisitId>
      </incomingNodes>
    </NodeTO>
    <NodeTO>
      <outgoingNodes>
        <int>9</int>
        <int>10</int>
      </outgoingNodes>
      <parameters>
        <Parameter>
          <maxIndex>0</maxIndex>
          <name>OPERAND_A</name>
          <pamValue>4</pamValue>
          <type>string</type>
          <action>NEW</action>
        </Parameter>
        <Parameter>
          <maxIndex>0</maxIndex>
          <name>OPERAND_B</name>
          <pamValue>5</pamValue>
          <type>string</type>
          <action>NEW</action>
        </Parameter>
        <Parameter>
          <maxIndex>0</maxIndex>
          <name>OPERATOR</name>
          <pamValue>GREATER</pamValue>
          <type>string</type>
          <action>NEW</action>
        </Parameter>
      </parameters>
      <processId>0</processId>
      <id>4</id>
      <nodeType>CONDITION</nodeType>
      <incomingNodes>
        <DataTypeVisitId>
```

```
          <Id>2</Id>
          <visited>false</visited>
        </DataTypeVisitId>
      </incomingNodes>
    </NodeTO>
    <NodeTO>
      <outgoingNodes />
      <parameters />
      <processId>0</processId>
      <id>6</id>
      <nodeType>CONDITION</nodeType>
      <incomingNodes />
    </NodeTO>
    <NodeTO>
      <outgoingNodes>
        <int>13</int>
      </outgoingNodes>
      <parameters>
        <Parameter>
          <maxIndex>0</maxIndex>
          <name>WS_TYPE</name>
          <pamValue>WEBSERVICE_1</pamValue>
          <type />
          <action>NEW</action>
        </Parameter>
        <Parameter>
          <maxIndex>0</maxIndex>
          <name>term</name>
          <pamValue>dog</pamValue>
          <type>string</type>
          <action>NEW</action>
        </Parameter>
      </parameters>
      <processId>0</processId>
      <id>9</id>
      <nodeType>TASK_NODE</nodeType>
      <incomingNodes>
        <DataTypeVisitId>
          <Id>4</Id>
          <visited>false</visited>
        </DataTypeVisitId>
```

```
      </incomingNodes>
    </NodeTO>
    <NodeTO>
      <outgoingNodes>
        <int>14</int>
      </outgoingNodes>
      <parameters>
        <Parameter>
          <maxIndex>0</maxIndex>
          <name>WS_TYPE</name>
          <pamValue>WEBSERVICE_2</pamValue>
          <type />
          <action>NEW</action>
        </Parameter>
        <Parameter>
          <maxIndex>0</maxIndex>
          <name>tax_id</name>
          <pamValue>9899</pamValue>
          <type>string</type>
          <action>NEW</action>
        </Parameter>
      </parameters>
      <processId>0</processId>
      <id>10</id>
      <nodeType>TASK_NODE</nodeType>
      <incomingNodes>
        <DataTypeVisitId>
          <Id>4</Id>
          <visited>false</visited>
        </DataTypeVisitId>
      </incomingNodes>
    </NodeTO>
    <NodeTO>
      <outgoingNodes />
      <parameters />
      <processId>0</processId>
      <id>13</id>
      <nodeType>END_NODE</nodeType>
      <incomingNodes>
        <DataTypeVisitId>
          <Id>9</Id>
```

```
          <visited>false</visited>
        </DataTypeVisitId>
      </incomingNodes>
    </NodeTO>
    <NodeTO>
      <outgoingNodes />
      <parameters />
      <processId>0</processId>
      <id>14</id>
      <nodeType>END_NODE</nodeType>
      <incomingNodes>
        <DataTypeVisitId>
          <Id>10</Id>
          <visited>false</visited>
        </DataTypeVisitId>
      </incomingNodes>
    </NodeTO>
  </_nodes>
</Structure>
```

Estructura JSON representando modelo de Imagen 26 - WEB GUI y modelo de Imagen 19 - Modelo de Flujo.

```
[{"name":"Example 1",
"working":
     "{\"modules\":
          [{\"config\":
               {\"position\":[15,154],
               \"xtype\":\"WireIt.InOutContainer\"},
          \"name\":\"Start\",
          \"value\":{}},
          {\"config\":
               {\"position\":[253,154],
               \"xtype\":\"WireIt.InOutContainer\"},
          \"name\":\"NCBI EUTILS\",
          \"value\":{}},
          {\"config\":
               {\"position\":[12,253]},
```

```
          \"name\":\"input\",
          \"value\":
               {\"input\":
                    {\"inputParams\":
                         {\"typeInvite\":\"input name\",
                         \"value\":\"cat\"},
                    \"type\":\"string\"}}},
          {\"config\":
               {\"position\":[538,154],
               \"xtype\":\"WireIt.InOutContainer\"},
          \"name\":\"Condition\",
          \"value\":{}},
          {\"config\":
               {\"position\":[269,377]},
          \"name\":\"input\",
          \"value\":
               {\"input\":
                    {\"inputParams\":
                         {\"typeInvite\":\"input name\",
                         \"value\":\"4\"},
                    \"type\":\"string\"}}},
          {\"config\":
               {\"position\":[-178,226],
               \"xtype\":\"WireIt.InOutContainer\"},
          \"name\":\"Condition\",
          \"value\":{}},
          {\"config\":
               {\"position\":[273,461]},
          \"name\":\"input\",
          \"value\":
               {\"input\":
                    {\"inputParams\":
                         {\"typeInvite\":\"input name\",
                         \"value\":\"5\"},
                    \"type\":\"string\"}}},
          {\"config\":
               {\"position\":[272,543]},
          \"name\":\"input\",
          \"value\":
               {\"input\":
                    {\"inputParams\":
```

```
                    {\"typeInvite\":\"input name\",
                    \"value\":\"GREATER\"},
               \"type\":\"string\"}}},
          {\"config\":
               {\"position\":[774,50],
               \"xtype\":\"WireIt.InOutContainer\"},
          \"name\":\"NCBI EUTILS\",
          \"value\":{}},
          {\"config\":
               {\"position\":[787,396],
               \"xtype\":\"WireIt.InOutContainer\"},
          \"name\":\"NCBI FETCHTAXON\",
          \"value\":{}},
          {\"config\":
               {\"position\":[513,341]},
          \"name\":\"input\",
          \"value\":
               {\"input\":
                    {\"inputParams\":
                         {\"typeInvite\":\"input name\",
                         \"value\":\"dog\"},
                    \"type\":\"string\"}}},
          {\"config\":
               {\"position\":[517,486]},
          \"name\":\"input\",
          \"value\":
               {\"input\":
                    {\"inputParams\":
                         {\"typeInvite\":\"input name\",
                         \"value\":\"9899\"},
                    \"type\":\"string\"}}},
          {\"config\":
               {\"position\":[1012,51],
               \"xtype\":\"WireIt.InOutContainer\"},
          \"name\":\"End\",
          \"value\":{}},
          {\"config\":
               {\"position\":[1025,395],
               \"xtype\":\"WireIt.InOutContainer\"},
          \"name\":\"End\",
          \"value\":{}}],
```

```
     \"properties\":
          {\"category\":\"Demo\",
          \"description\":\"NCBI-EUTILS or NCBI-FETCHTAXON\",
          \"isTest\":false,
          \"name\":\"Example 1\"},
     \"wires\":
          [{\"src\":{\"moduleId\":0,\"terminal\":\"SOURCES\"},
          \"tgt\":{\"moduleId\":1,\"terminal\":\"FOLLOWUPS\"}},
          {\"src\":{\"moduleId\":2,\"terminal\":\"out\"},
          \"tgt\":{\"moduleId\":1,\"terminal\":\"term\"}},
          {\"src\":{\"moduleId\":1,\"terminal\":\"SOURCES\"},
          \"tgt\":{\"moduleId\":3,\"terminal\":\"FOLLOWUPS\"}},
          {\"src\":{\"moduleId\":4,\"terminal\":\"out\"},
          \"tgt\":{\"moduleId\":3,\"terminal\":\"OPERAND_A\"}},
          {\"src\":{\"moduleId\":6,\"terminal\":\"out\"},
          \"tgt\":{\"moduleId\":3,\"terminal\":\"OPERAND_B\"}},
          {\"src\":{\"moduleId\":7,\"terminal\":\"out\"},
          \"tgt\":{\"moduleId\":3,\"terminal\":\"OPERATOR\"}},
          {\"src\":{\"moduleId\":3,\"terminal\":\"SOURCES\"},
          \"tgt\":{\"moduleId\":8,\"terminal\":\"FOLLOWUPS\"}},
          {\"src\":{\"moduleId\":10,\"terminal\":\"out\"},
          \"tgt\":{\"moduleId\":8,\"terminal\":\"term\"}},
          {\"src\":{\"moduleId\":3,\"terminal\":\"SOURCES2\"},
          \"tgt\":{\"moduleId\":9,\"terminal\":\"FOLLOWUPS\"}},
          {\"src\":{\"moduleId\":11,\"terminal\":\"out\"},
          \"tgt\":{\"moduleId\":9,\"terminal\":\"tax_id\"}},
          {\"src\":{\"moduleId\":8,\"terminal\":\"SOURCES\"},
          \"tgt\":{\"moduleId\":12,\"terminal\":\"FOLLOWUPS\"}},
          {\"src\":{\"moduleId\":9,\"terminal\":\"SOURCES\"},

     \"tgt\":{\"moduleId\":13,\"terminal\":\"FOLLOWUPS\"}}]}",
"language":"meltingpotDemo"}]
```


9.3 Trabajos Relacionados

A continuación se enumeran en forma resumida las 5 librerías mencionadas en la sección 2.3.6 – Microsoft Biology Foundation:

1. MumUtil – Maximal Unique Match

MumUtil encuentra coincidencias únicas entre un archivo de referencia y uno o más archivos de entrada utilizando árboles de sufijos.

Dado un conjunto de secuencias, una coincidencia única es una coincidencia que ocurre únicamente una vez en cada secuencia. Luego una maximal unique match (MUM), es una coincidencia que no es parte de ninguna coincidencia mayor.

La herramienta maneja archivos en formato fasta y requiere la definición de parámetros como ser búsqueda en reversa, mínimo largo de la máxima coincidencia, habilitación de logs, entre otros.

2. NucmerUtil – Multi Alignment Tool

La herramienta NucmerUtil permite el alineamiento de múltiples referencias y secuencias utilizando la herramienta MUMmer. Esta última se encuentra principalmente dirigida a largas secuencias de ADN.[Ref21][Ref22]

Nucmer genera alineamiento de nucleótidos entre dos archivos de entrada multi-FASTA. De este alineamiento, dos archivos son generados. El primer archivo, de extensión .cluster, lista conjuntos o clústers de coincidencias entre ambas secuencias. Por el contrario, el segundo archivo de salida de extensión .delta, lista la distancia entre inserciones y "deleciones" que producen una máxima coincidencia al alinear las secuencias.

Esta herramienta permite configurar si se realizan alineamientos en reversa o únicamente en forma directa, si se usan puntos de anclaje al realizar el alineamiento, y si los mismos pueden estar en ambas secuencias o en solo una de ellas, etc..

3. SAMUtils – SAM <-> BAM conversion

La herramienta SAMUtills trabaja con archivos de mapas de alineamiento de secuencia y permite la conversión entre los formatos SAM y BAM. El formato SAM se denomina Sequence Alignment Map y es el formato en el que el científico puede leer la información de alineamiento de secuencias. Por el contrario, el formato BAM es el formato Binary Alignment Map y es el formato comprimido del anterior.

En esta herramienta se pueden configurar opciones tales como la creación de índices o la unión de múltiples archivos en el mismo formato a uno único.[Ref23]

4. ComparativeUtil – Sequence Assembly

La herramienta ComparativeUtil ejecuta comparación de secuencias con armado de nuevas secuencias a partir de las mismas.

La herramienta usa como entrada otras secuencias cercanas como referencia y combina otras herramientas de más bajo nivel como ser: MUMUtil para ejecutar las comparaciones, NucUtil para realizar los alineamientos, LayoutRefinementUtil para ajustar la distribución de los alineamientos, RepeatResolutionUtil para eliminar la ambigüedad o incertidumbre en la ubicación de las nuevas secuencias resultado del alineamiento y ConcensusUtil para reducir aun más la incertidumbre y generar un consenso general de donde se deben ubicar las secuencias en el alineamiento resultado. Por último, la presente herramienta utiliza ScaffoldUtil para unir todas las secuencias resultado.

Se pueden configurar opciones como desviación estándar, utilización de librerías adicionales, ajuste de largo de secuencias y salteo de pasos.

5. DeNovo Assembly

La herramienta PadenaUtil permite el armado de secuencias para la generación de contigs (conjunto de segmentos de ADN que se solapan y que en su totalidad representan una zona de consenso del ADN) a partir de múltiples lecturas. No requiere de diferencia de la anterior de un alineamiento referencia.