Centro de Investigación y de Estudios Avanzados
del Instituto Politécnico Nacional

Departamento de Computación

Tavlab: entorno estadístico basado en *workflows* con Taverna

Tesis que presenta

Ulises Revilla Duarte

para obtener el Grado de

Maestro en Ciencias

en la Especialidad de

Ingeniería Eléctrica

Codirectores de la Tesis

Dr. Jorge Buenabad Chávez
Dra. Ana María Antonia Martínez Enríquez

México, D.F. **Marzo 2008**

Resumen

Esta tesis presenta el diseño e implementación de una infraestructura de componentes de software distribuidos que pueden integrarse en sistemas de *workflows* destinados a la resolución de problemas científicos de análisis estadístico de datos experimentales. Concretamente, esta infraestructura permitirá al usuario final interactuar con flujos de datos dentro de un ambiente gráfico de solución de problemas en forma distribuida (DPSE: Distributed Problem Solving Environment). En un ambiente gráfico de este tipo, se utilizan los recursos de múltiples equipos individuales interconectados mediante una red (principalmente la Internet) para la solución de problemas computacionales a gran escala. Con la finalidad de validar la infraestructura de servicios Web construida, se resuelven diversos casos prácticos en Biotecnología, Ingeniería Química y Medicina Estadística.

Abstract

This thesis presents the design and implementation of an infrastructure of software components to be integrated within workflows for statistical analysis of experimental data. This infrastructure allows users to interact with data flows within a graphical problem-solving environment in a distributed way, if need be. This kind of environment makes possible to use multiple distributed resources reachable through the Internet in the solution of large-scale computational problems. To validate our infrastructure, several practical problems from Biotechnology, Chemestry Engineering and Statistical Medicine were solved.

Agradecimientos

Al **Centro de Investigación y de Estudios Avanzados del I.P.N.** por abrirme las puertas de su importante centro de investigación, y permitirme seguir con mis estudios de Maestría.

Al **Consejo Nacional de Ciencia y Tecnología** por la beca que me otorgó para realizar mis estudios de Maestría.

Al **Dr. Jorge Buenabad Chávez** por su amistad, invaluable apoyo y consejos, e infinita paciencia y guía en el desarrollo de la presente tesis.

A la **Dra. Ana María Antonia Martínez Enríquez** por su amistad, por dirigir mi trabajo de investigación con enorme interés y entusiasmo, por su invaluable apoyo y consejos, así como por el impulso decisivo para la terminación de mi documento de tesis.

Al **Dr. Adriano de Luca Pennacchia** por su apoyo e interés en mi trabajo de investigación y por su participación como parte del Jurado en el examen de Grado.

Al **Dr. José Oscar Olmedo Aguirre** por la revisión de mi documento de tesis, por contribuir al mejoramiento de la misma a través de sus comentarios y preguntas, así como por su participación como parte del Jurado en el examen de Grado.

A la **Dra. Sonia Mendoza Chapa** por su gran apoyo y consejos, por sus palabras de motivación e interés en la terminación de este trabajo de tesis.

Al **Dr. Sergio Chapa Vergara** por su apoyo y consejos.

A los **ingenieros Mirna Guadalupe Botello Sandoval e Iván Israel Uresti Adame** por su amistad, por la revisión de mi documento de tesis y por el apoyo que siempre me han brindado.

A mis **compañeros de la generación 2005** por el apoyo que me brindaron durante la Maestría.

A la **Sra. Sofía Reza Cruz** quien amablemente me orientaba sobre los tramites administrativos durante mi estancia en el Departamento de Computación.

Así, como a todas las personas que directa o indirectamente han contribuido para la realización de esta tesis.

A todos Ustedes muchas gracias.

Dedicatoria

A **Dios** que siempre me ayuda y me da fuerzas para seguir esforzándome.

A la memoria de mi padre,
Javier Miguel Revilla Arellano

A mi madre,
Mercedes Duarte Prado

A mis hermanos,
Aquiles y Benjamín Revilla Duarte

A mi abuela **Elena**

A mi tío,
Sergio Duarte Prado y a su esposa **Adriana**

A mis sobrinos,
Javier, **Ulises** y **Jesús Alejandro**.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

Recientemente, el poder de cómputo y los algorítmos informáticos para resolver problemas científicos han alcanzado un desarrollo considerable aportando contribuciones significativas a la *e-Science*[1]. Así también, con la proliferación de la Internet, la solución de problemas científicos puede llevarse a cabo satisfactoriamente compartiendo recursos de cómputo distribuidos. Al mismo tiempo, los sistemas de *workflows* [2] cientifícos proporcionan una representación de flujos de datos dentro de un ambiente distribuido de solución de problemas DPSE (*Distributed Problem Solver Environment*) bastante atractiva para análisis de datos.

Por su parte, la computación Grid[3] busca cambiar la manera en la cual los científicos acceden recursos de cómputo distribuidos por medio de la creación de una infraestructura orientada a servicio para aplicaciones de Computación Científica [19].

Un ambiente DPSE consta de un conjunto completo y conveniente de herramientas para resolver problemas pertenecientes a un dominio específico. DPSE permite: a) definir y modificar problemas, b) escoger estrategias de solución, c) interactuar adecuadamente con recursos de hardware y software, así como d) visualizar y analizar resultados. Un usuario se comunica con un DPSE en el lenguaje propio del problema, no en el lenguaje de un sistema operativo particular, lenguaje de programación o protocolo de red [82].

DPSE hace uso la tecnología más avanzada en muchas subdisciplinas de las Ciencias de la Computación para crear ambientes robustos y aplicarlos en áreas específicas, principalmente, en la ciencia, la ingeniería y la manufactura. Las subdisciplinas relevantes son: Computación Distribuida y Paralela, Computación Colaborativa, Graficación y Visualización, Interacción Hombre-Computadora, Redes Computacionales y la Web, Programación Orientada a Objetos, Inteligencia Artificial e Ingeniería de Software [82].

[1] El término e-Science (o eScience) se usa para describir procesos científicos demandantes de computación intensiva, la cual es llevada a cabo a través de redes altamente distribuidas.

[2] Término traducido frecuentemente en la literatura informática como diagrama de flujo o flujo de trabajo

[3] La Grid es una infraestructura compuesta de un conjunto de recursos computacionales potencialmente compartidos, distribuidos, heterogéneos y autónomos

Los principales ambientes de *workflows* para resolución de problemas científicos actualmente en pleno desarrollo son (Capitulo 2):

1) **Triana**[77]: realiza tareas de procesamiento de señales, de texto y de imágenes.

2) **Ptolemy**[81]: modela, simula y diseña sistemas concurrentes, de tiempo real y embebidos.

3) **Kepler**[78]: se utiliza para en diseño de *workflows* en las áreas de Astrofísica, Ecología y Geología. Se basa en el proyecto Ptolemy.

4) **SciRUN**[79]: realiza análisis de imágenes biomédicas.

5) **Taverna**[41]: facilita el uso y la integración del creciente número de bases de datos y herramientas bioinformáticas en la Web.

1 Agentes de software

En la Web, así como en ambientes dedicados, el paradigma de sistemas distribuidos basados en agentes está evolucionando. Los agentes son instancias activas que pueden moverse a través de la red y ser ejecutados en una localidad diferente a su sitio original. Los agentes ofrecen ventajas que idealmente corresponden a los requerimientos de gestión descentralizada y flexible de *workflows*. Los agentes reducen el tráfico en la red, puesto que se ejecutan donde están los recursos, esto es, en un escenario de *workflow* el control necesario toma lugar en la máquina del cliente y no mediante una interacción con el servidor central. La descentralización heredada, consigue una gran robustez en la red y disminuye las fallas en el servidor. La tecnología de agentes permite desacoplar en gran medida a los clientes del servidor de *workflows* [11].

La Inteligencia Artificial Distribuida [12] es un subcampo de la IA dedicada al estudio de las técnicas y el conocimiento necesario para la coordinación y distribución de las acciones en un entorno con múltiples agentes. La Resolución Cooperativa de Problemas Distribuidos (CDPS, Cooperative Distributed Problem Solving) es una área de la IAD que estudia cómo un conjunto de agentes cooperan para dividir y compartir tareas acerca de una problemática y del desarrollo de la solución. A continuación, se describen los sistemas basados en componentes (o agentes) de software distribuidos.

1. Descripción, descomposición y asignación de tareas: la descripción debe incluir la información sobre las características y los atributos del problema así como del dominio y del entorno del problema. Una vez dada la descripción de una tarea, la descomposición de ésta y asignación de subtareas a múltiples agentes debe tener en cuenta la capacidad y recursos disponibles de los agentes para realizarla.

Bond y Gasser [13,14] han identificado varias dimensiones utilizadas comúnmente para realizar la descomposición de un problema:

- Nivel de abstracción: los agentes que actúan para resolver problemas pueden ser asociados a un nivel de abstracción del problema. La descomposición del problema en niveles de abstracción proporciona una buena base para la descomposición de tareas [15,16].
- Control y dependencias: las tareas se descomponen tomando como principio la reducción de las dependencias de datos o control entre tareas, de forma que se limite la comunicación. La dependencia de los datos puede ser semántica, temporal o con respecto a la distribución de los datos de entrada. Por ejemplo, en una red de sensores [15], cuando se distribuye una tarea, suele ser necesario introducir tareas de control para coordinarla.
- División funcional/producto: la descomposición funcional agrupa a los componentes que realizan funciones similares, mientras que la división por productos agrupa a los que trabajan en la producción de un mismo producto.
- Para evitar cuellos de botella, debe evitarse que una tarea sea resuelta por un sólo componente.
- Optimización de recursos: con la finalidad de evitar sobrecargas de comunicación y coordinación, es necesario una buena gestión de los recursos compartidos.

Las técnicas más empleadas para dividir automáticamente las tareas por parte de los agentes son:

- Tareas inherentes divisibles: la propia descripción de la tarea incluye en sí su descomposición.
- Descomposición por el programador.
- Planificación jerárquica: las tareas se definen en forma de planes que satisfacen objetivos y dan lugar a árboles Y-O. Un plan contiene subplanes.
- Agregación de subtareas: enfoque ascendente en vez de enfoque descendente en la descomposición.

2. Comunicación: los agentes mejoran su coordinación gestionando qué, cómo y cuándo deben comunicarse entre sí [13]. Mediante la comunicación, los agentes adquieren el conocimiento necesario para actuar con una visión no sólo local, sino con un alcance mayor y sincronizarse con todos los agentes que integran el grupo. Sin embargo, una comunicación excesiva puede dar lugar a que la sobrecarga de comunicación sea mayor que el trabajo efectivo realizado. Se distingue un amplio rango de formas de comunicación [16], que van desde la ausencia hasta la comunicación de alto nivel y la interacción hombre máquina.

Por otra parte, los agentes de software que componen un *workflow* presentan relaciones temporales [38]. Estas relaciones determinan el curso del flujo de datos dentro del *workflow*, esto es, la manera en que éste se ejecuta. A continuación, se describen estas relaciones temporales y se muestran en la *Figura 1.1*.

(a) Secuencia: se presenta cuando un componente debe ejecutarse después que el anterior se ha completado. En nuestro ejemplo, el componente **B** puede ejecutarse sólo cuando el componente **A** ha completado su ejecución y, a la vez, **B** debe de concluir antes de que el componente **C** pueda activarse.

(b) Concurrencia: requiere que varios componentes se ejecuten concurrentemente. Ambas tareas **B** y **C** se activan en el momento en que el componente **A** termina su ejecución.

(c) Sincronización: requiere que varias actividades concurrentes terminen para dar inicio a otra nueva actividad. En nuestro ejemplo, el componente **C** se ejecuta sólo después que los componentes **A** y **B** terminan.

(d) Selección exclusiva: se presenta cuando después de haberse completado la ejecución de un componente **A**, ya sea **B** o **C** pueden ser activados.

(e) Empalme: se presenta cuando alguno de varios componentes que se ejecutan concurrentemente termina primero; en el ejemplo **C** se ejecuta cuando **A** o **B** terminan.

(f) Disyunción: se construye para escoger varios alternativas de un conjunto. En nuestro ejemplo, una vez que **A** terminó, se podría ejecutar **B** o **C** o ambos.

(g) Combinación múltiple: permite la activación múltiple de un componente y no requiere sincronización después de la ejecución de tareas concurrentes. Una vez que **A** termina, los componentes **B** y **C** se ejecutan concurrentemente. Cuando alguno de ellos termina primero, digamos **B**, entonces el componente **D** se activa; luego, en el momento en que **C** termina, **D** es activado nuevamente.

(h) Discriminación: espera a que un número de componentes termine antes de ejecutar al siguiente; luego, espera a que el resto de los componentes terminen sin ejecutar ninguna acción.

(i) Barrera de sincronización: supóngase que se cuenta con **M** componentes de los cuales **N** ($N < M$) se ejecutan concurrentemente y tienen que alcanzar la barrera antes que la siguiente tarea sea activada. En nuestro ejemplo, cualesquiera 2 de los 3 componentes **A**, **B**, o **C** deben de terminar antes de poder ejecutar **E**.

(j) Aplazamiento: es similar a la selección exclusiva del inciso (d) pero esta vez la selección no es explícita y las condiciones de ejecución del *workflow* determina qué rama tomar.

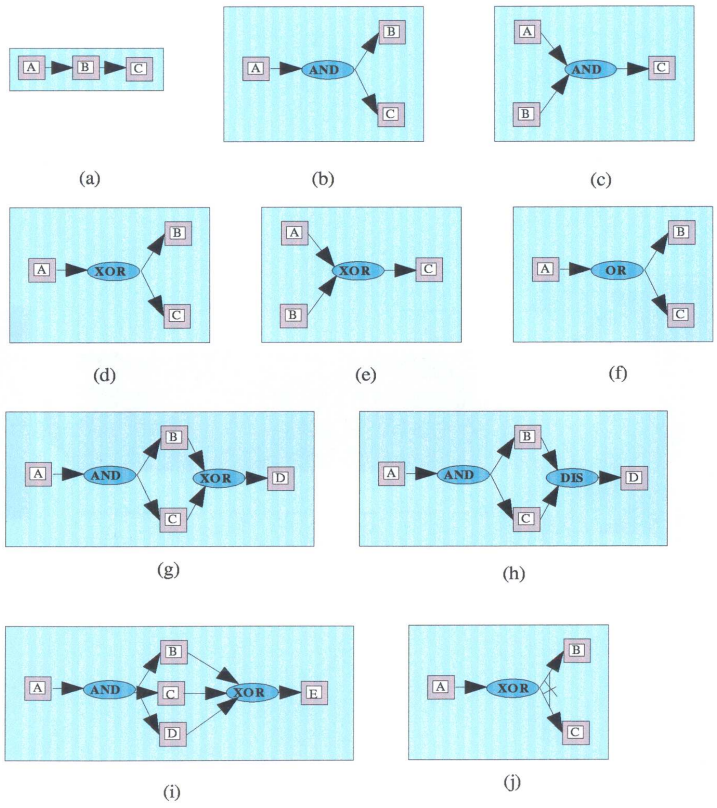


Figura 1.1: Relación temporal de un *workflow*

2 Modelo de un *workflow*

Un agente es un programa que realiza una función de control [38].

La *Figura 1.2* muestra un modelo donde:

(i) El motor del sistema de *workflows* es un agente responsable de coordinar su funcionamiento.

(ii) Cada componente individual es ejecutado bajo la supervisión de un agente de control.

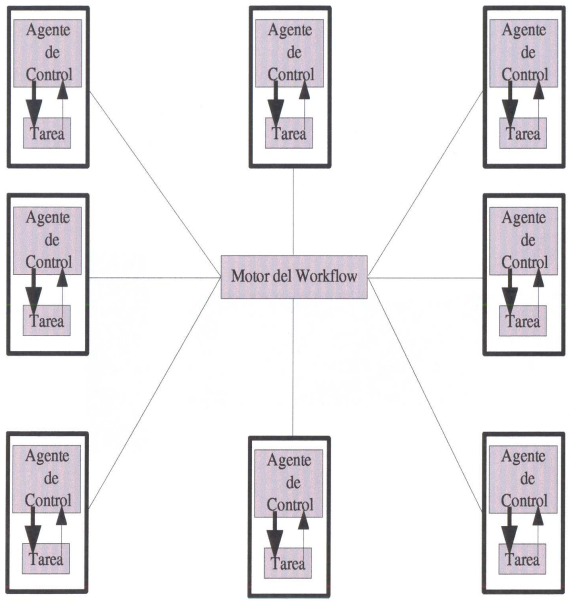


Figura 1.2: Coordinación del *workflow* a través del motor de nivel superior

El aspecto de la coordinación del funcionamiento del *workflow* abarca tanto la capa de proceso como la de recursos. El modelo presentado en la *Figura 1.2*, delega la supervisión de las tareas específicas de cada componente a los agentes de control. Esta estrategia mantiene el tráfico de la red al mínimo puesto que las decisiones específicas a cada tarea se toman localmente reduciéndose el tiempo para completarlas.

Este modelo facilita el diseño de motores genéricos de *workflows* y promueve una actuación jerárquica con múltiples motores coordinando diferentes aspectos del *workflow*.

Una actuación jerárquica de *workflow* es más escalable (ver la *Figura 1.3*). El proceso de toma de decisiones descentralizado limita el tráfico y disminuye el tiempo de decisión. En este modelo, el motor del nivel superior provee el proceso de coordinación total y coopera con los motores independientes para la asignación de recursos.

La *Figura 1.3*, muestra dos niveles de jerarquía consistentes en un motor de nivel superior y varios motores individuales que coordinan la ejecución de un subconjunto de componentes. Para *workflows* más complejos se consideran jerarquías multicapa.

La *Figura 1.4*, muestra el ciclo de vida de los agentes involucrados en un modelo de *workflow* jerárquico para un caso que incluye cinco tareas, A, B, C, D y E,

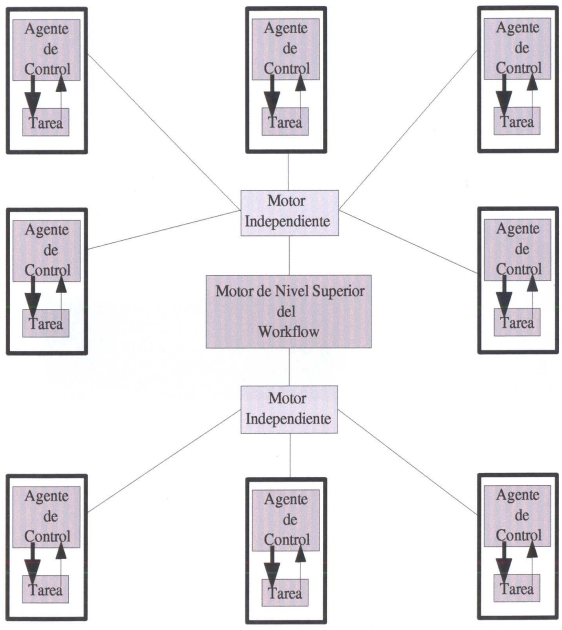


Figura 1.3: Coordinación jerárquica del *workflow*

dos motores independientes, AB y CDE, y un motor de nivel superior. El motor de nivel superior del sistema de *workflow* se activa en el tiempo t_0 y, a la vez, dispara la ejecución de los dos motores independientes en los tiempos t_1 y t_4, en respuesta a un evento de activación de la primera tarea en cada grupo. Las máquinas independientes son desactivadas por el motor de nivel superior cuando todas las tareas se han completado en los tiempos t_5 y t_7 respectivamente.

Taverna

Taverna es un ambiente distribuido gráfico de resolución de problemas perteneciente a *myGrid Project* [4]. Taverna permite construir *workflows* de procesos complejos combinando componentes localizados tanto en máquinas locales como remotas, ejecutar estos *workflows* con sus propios datos y visualizar los resultados. Un componente de software es una función, un procedimiento o un método ofrecido por un servicio Web. En el ambiente de Taverna, cada componente de software se representa por medio de un ícono reusable etiquetado con el nombre de la función que realiza. Los servicios Web son aplicaciones distribuidas accesibles desde cualquier sitio Web. De

[4] *myGrid* es el proyecto dentro de la *e-Science* que da soporte a las diferentes plataformas de cómputo en las cuales se realizan diversas aplicaciones científicas, como el diseño y la ejecución de *workflows*, así como la administración de datos y metadatos.

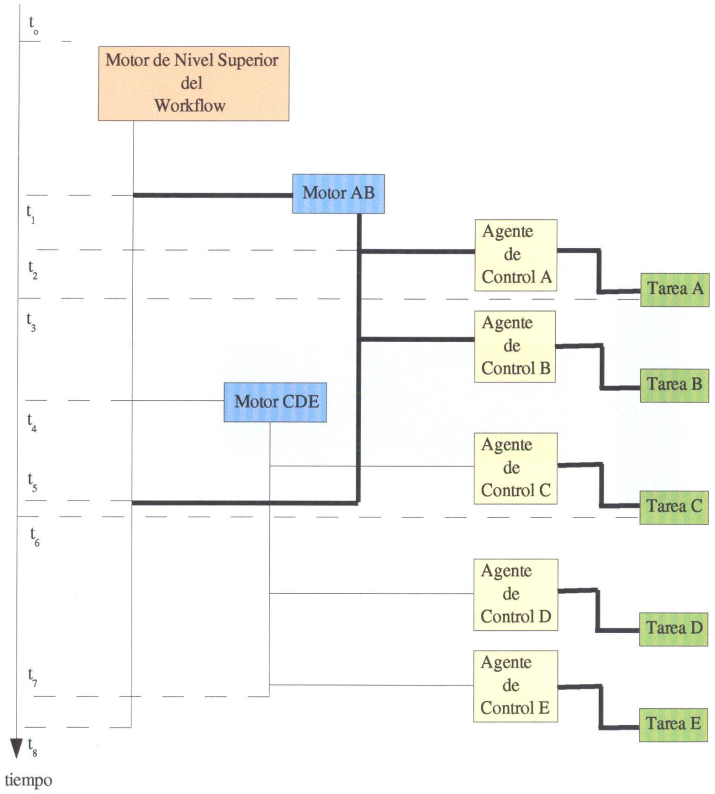


Figura 1.4: Ciclo de ejecución de un *workflow* jerárquico

esta manera, un *workflow* es un diagrama de flujo construido mediante la conexión de las entradas y salidas de estos íconos. Taverna es un proyecto de software abierto que permite principalmente: programar en forma gráfica a científicos (que no cuentan necesariamente con experiencia en programación), compartir e integrar recursos diferentes en un solo análisis, realizar varias aplicaciones al mismo tiempo y añadir acciones de coordinación a las aplicaciones (Capítulo 2).

Taverna se concibió en un principio para facilitar el uso y la integración del creciente número de bases de datos y herramientas bioinformáticas en la Web [2]. Por lo que existe una amplia gama de servicios disponibles para la Bioinformática (por ejemplo, para obtener alineamientos de proteínas, secuencias de ADN, etc). Algunos de los organismos que proveen estos servicios son el INSDC (International Nucleotic Sequence Database Collaboration), el EMBL-EBI (European Bioinformatics Institute), el NCBI (National Center for Biotechnology Information) y el DDBJ (DNA Databank of Japan). Sin embargo, recientemente, investigadores de otras disciplinas han encontrado en Taverna una herramienta gráfica atractiva en la cual desarrollar aplicaciones. Tal es el caso de la Quimioinformática y la Imagenología Médica [1][2].

La primera versión de Taverna aparece disponible en el mes de febrero de 2003 como un proyecto orientado a satisfacer los requerimientos de biólogos en la comunidad científica del Reino Unido, con la participación del Instituto Europeo de Bioinformática (EMBL-EBI) y de varias universidades inglesas, principalmente la Universidad de Manchester [41]. Taverna provee un entorno independiente del Sistema Operativo. Para febrero de 2006, el número de descargas de Taverna alcanzó la cifra de 14,000 [2]. Para febrero del 2007, esta cifra aumentó a 31,747 [41]. En sólo un año se incrementó el 126.7 %. La versión actual de Taverna (julio del 2007) es la 1.5.2.

3 Motivación y justificación

La motivación principal para este trabajo de tesis consiste en desarrollar una infraestructura de componentes de software para tratar datos obtenidos experimentalmente con un enfoque diferente al que se ha usado tradicionalmente. Una infraestructura para Computación Científica aplicada a la resolución de problemas de análisis estadístico de datos experimentales que aproveche recursos de cómputo distribuidos y que permita interactuar con flujos de datos en un entorno gráfico. Esto presenta varias ventajas respecto a otras herramientas o paquetes de software que realizan cálculos estadísticos (como Stata[5], SAS[6], Minitab[7], GenStat[8], etc), por ejemplo, la visualización de todos los componentes del proceso solución, así como del comportamiento general del mismo. Taverna no se limita al simple hecho de crear diagramas, sino que es capaz de asociar funcionalidades a cada nodo del diagrama de flujo de manera que ejecute realmente cada paso del proceso. Nuestra infraestructura permite diseñar, ejecutar y compartir *workflows* de análisis estadísticos como un nuevo enfoque de resolución de problemas. Toda esta infraestructura tiene como finalidad incrementar la robustez de la capa de recursos dentro de la arquitectura de capas de la plataforma Taverna, poniendo a disposición del usuario científico servicios Web para el análisis estadístico de datos experimentales. De esta manera, el científico sólo necesita saber las direcciones de Internet URL (Uniform Resource Lacator) de los servicios Web con las aplicaciones que requiere para construir el *workflow* de un experimento o análisis. La ejecución del *workflow* es transparente para el usuario.

4 Planteamiento del problema

La aplicación propuesta se enfoca en el tratamiento de datos suministrados por científicos de diversas áreas para encontrar la solución de problemas de análisis de datos utilizando recursos de software distribuidos en la Web. Taverna proporciona la tecnología que nos permite, precisamente, conjuntar los recursos de software reque-

[5] http://www.stata.com
[6] http://www.sas.com
[7] http://www.minitab.com
[8] www.vsni.co.uk/products/genstat/

ridos en un entorno gráfico. Utiliza los recursos de múltiples equipos individuales, interconectados mediante una red (principalmente la Web) para la solución de problemas computacionales a gran escala. Esto significa que nuestra implementación deberá ser capaz de aprovechar las ventajas que supone disponer de múltiples computadoras interconectadas, modelando una arquitectura de ordenador virtual, que nos permite distribuir la ejecución de los procesos a lo largo de estas infraestructuras paralelas. Fundamentalmente, la idea consiste en tener servicios en la Web definidos y organizados de tal manera que nos facilite su descubrimiento, integración y reutilización en múltiples aplicaciones.

Objetivo

Esta tesis tiene como objetivo, la implemetación de un entorno de servicios Web con componentes de software diseñados para realizar aplicaciones de análisis estadístico y de análisis de varianza de datos experimentales, los cuales pueden ser integrados, reutilizados y compartidos para construir *workflows* en la plataforma Taverna.

Estos servicios Web ofrecen componentes de software distribuidos para realizar análisis estadístico de datos experimentales, aprovechando las ventajas de múltiples computadoras interconectadas a través de una red, especialmente la Internet, para realizar múltiples cálculos simultáneamente mediante la división de tareas en la plataforma gráfica de Taverna. Tales recursos consistirán de procedimientos para obtener, por ejemplo: las medidas de tendencia central y de dispersión, la ecuación de regresión y su gráfica, el coeficiente de correlación, la prueba t de Student, etc.

Con la finalidad de validar nuestra implementación, resolvemos problemas de análisis de datos en diversas disciplinas, primero de manera conceptual (desde un nivel de abstracción específico al tipo de problema) y en seguida, por medio de nuestra aplicación para mostrar su funcionalidad, así como sus ventajas.

Metodología

Para cumplir con nuestro objetivo, llevamos a cabo la siguiente metodología:

1. **Diseño e implementación de funciones de transformación de datos en cualquier formato.**

   La meta en esta fase, es diseñar y crear servicios que permitan la lectura de archivos de datos de entrada en diversos formatos y en cualquier disposición. Por ejemplo: archivos con matrices de datos, archivos binarios en formato Integer y sistema Big Endian, etc (Capitulo 3).

2. **Desarrollo de métodos estadísticos que sean invocables desde Taverna.**

   En esta fase nuestra meta es el diseño y desarrollo de servicios destinados a la resolución de problemas de análisis estadístico de datos característicos en diversos campos de la ciencia. Los métodos que incluirán estos servicios serán, entre

otros, aquellos para calcular: las medidas de tendencia central y de dispersión, el coeficiente de correlación, la varianza, la varianza conjunta, la ecuación de regresión, el error estimado para la pendiente, la prueba t de Student para la varianza conjunta y la correlación, etc (Capítulos 4 y 5).

3. **Diseño e implementación de un método para graficar la línea de regresión.**

   Se implementará un servicio que permitirá graficar la ecuación de regresión, así como la dispersión de los datos de entrada (Capítulo 5).

4. **Desarrollo de los métodos para el análisis de varianza de múltiples medias.**

   Hemos desarrollado servicios con los métodos característicos para realizar análisis de varianza de múltiples medias obtenidas a partir de datos experimentales. Estos métodos son: el método de Tuckey y el método de la diferencia significativa mínima LSD (*Less Significant Difference*) de Fisher (Capítulo 4).

5 Contribución

Nuestras contribuciones son servicios Web agrupados en 4 niveles principales de acuerdo a la aplicación para la cual fueron diseñados. A su vez, cada servicio Web cuenta con varios métodos de objetivo específico. Estos niveles forman un entorno de servicios Web al que hemos llamado: Tavlab (Capítulo3). Tavlab se muestra en la *Figura 1.5*.

- El nivel I agrupa los servicios Web diseñados para recibir archivos de datos experimentales. Estos servicios cuentan con métodos para lectura de datos de entrada sin importar su disposición y codificación (ascii o binaria) en el archivo, y los presentan a la entrada de un *workflow* en forma adecuada (arreglo tipo string de datos) para su procesamiento.

- En el nivel II se encuentran los servicios Web diseñados con métodos propios de la Estadística Descriptiva y del análisis de varianza de datos experimentales.

- En el nivel III se crearon servicios Web con métodos para realizar las operaciones aritméticas de la raíz cuadrada, elevar al cuadrado y valor absoluto. Estas operaciones se utilizan con frecuencia en los cálculos estadísticos, por ejemplo, la desviación estándar se obtiene al aplicar la raíz cuadrada a la varianza. Esta organización permite que la construcción de *workflows* estadísticos sea más clara y directa. En este mismo nivel III, se incluyeron los servicios que implementan las tablas estadísticas empleadas en los cálculos de análisis de varianza y las pruebas de Tukey y de la diferencia significativa mínima LSD (*Less Significant Difference*) de Fisher.

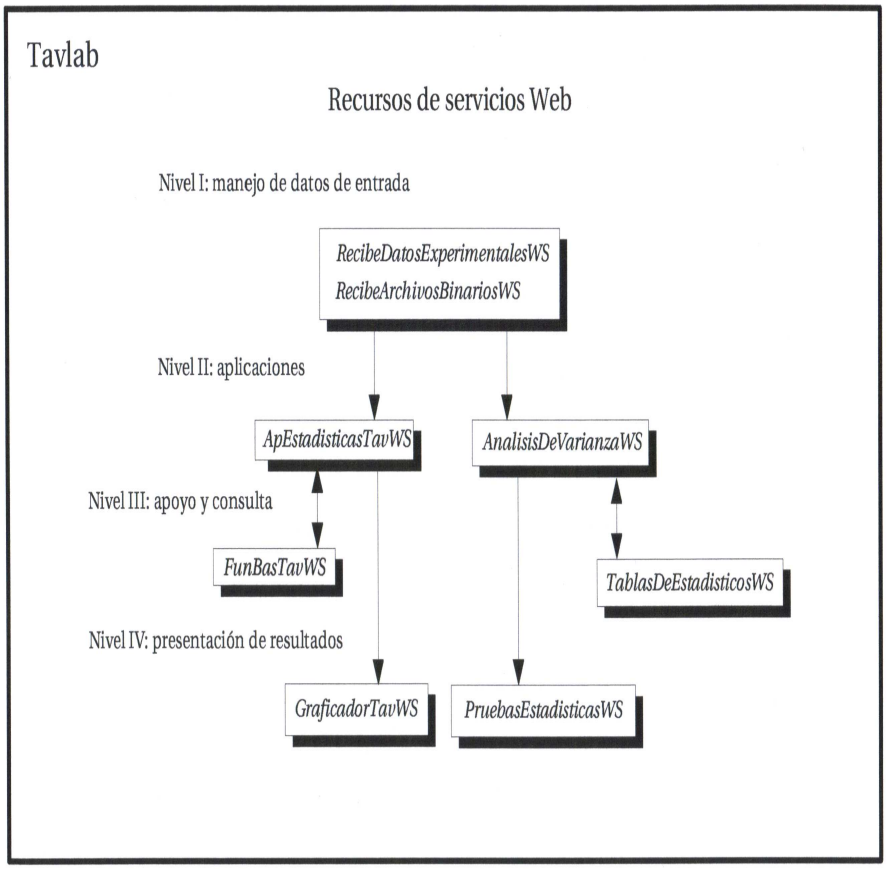


Figura 1.5: Recursos de servicios Web en Tavlab

- En el nivel IV se encuentran los servicios destinados a graficar la línea de regresión sobre la dispersión de los datos, así como las tablas de resultados correspondientes al análisis de varianza y las pruebas de Tukey y de la diferencia significativa mínima LSD (*Less Significant Difference*) de Fisher.

Para implementar los servicios Web del entorno Tavlab se requirió conocer y aplicar los estándares abiertos (Capítulo 3):

1) Lenguaje de marcado extensible XML (*eXtensive Markup Language*): estructura la información que se envía.

2) Protocolo simple de acceso a objetos SOAP (*Simple Object Access Protocol*): permite intercambiar los mensajes entre el cliente y el servicio Web.

3) Descriptor de activación de servicios WSDD (*Web services Deployment Descriptor*): activa los componentes de software (métodos) ofrecidos por los servicios Web.

4) Lenguaje de descripción de servicios WSDL (*Web Services Description Language*): describe los componentes de software ofrecidos por los servicios Web.

También fue necesario instalar un servidor Apache-Axis para utilizarlo como contenedor de servicios Web. La instalación de Apache-Axis y Taverna se describe en el Apéndice A.

Para realizar los programas de las aplicaciones estadísticas ofrecidas por los servicios Web fue indispensable conocer con profundidad los paradigmas de la programación orientada a objetos, en particular del lenguaje Java. Un aspecto importante de nuestra investigación se centró, precisamente, en la programación en lenguaje Java de aplicaciones que generaran gráficas para mostrarlas en Taverna.

Para mostrar cómo utilizar nuestro entorno Tavlab se resolvieron problemas de análisis estadístico y de análisis de varianza de datos experimentales en los campos de la Biotecnología, la Quimiometría y la Medicina Estadística (Capítulos 4 y 5).

La Biotecnología es el campo del conocimiento que integra el uso de las ciencias de la Bioquímica y la Microbiología con la ingeniería para llevar acabo aplicaciones controladas y deliberadas de las capacidades de agentes biológicos, (microorganismos, cultivos celulares, etc) para uso industrial [24].

La Quimiometría es la disciplina química que se enfoca en la aplicación de métodos matemáticos o estadísticos sobre datos químicos [27].

La Medicina Estadística se apoya en métodos estadísticos para la investigación de los mecanismos de la enfermedad (Patología Humana) y del rendimiento de los procedimientos de diagnóstico y tratamiento [20].

Caso de estudio:

Para la Ingeniería Química, la cinética de una reacción debe ser conocida si se desea hacer un diseño satisfactorio para equipo en el cual esas reacciones sean eficientes en la escala industrial. Por supuesto, si la reacción es lo bastante rápida para que el sistema esté esencialmente en equilibrio, el diseño es muy simplificado y solamente la información termodinámica es suficiente.

El estudio de la cinética de una reacción comienza con la determinación de la velocidad de reación, para más adelante llevar a cabo el estudio del mecanismo de la velocidad de reacción. La predicción y conocimiento de esta última se deduce a partir de varios modelos posibles, estableciendo las ecuaciones cinéticas globales y comprobando si se ajustan a los resultados experimentales. De tal suerte que la Cinética Química puramente teórica está limitada por el grado de certeza de la estimación de los resultados obtenidos experimentalmente.

A continuación, se muestra cómo se utilizarán los componentes de software de esta infraestructura mediante la resolución de un problema en Ingeniería Química. Se presenta primero su descripción, después su solución tradicional, y en seguida, su solución bajo el enfoque de *workflow*.

Ejemplo:

En una serie de experimentos para la determinación de estaño en productos alimenticios, las muestras fueron llevadas a ebullición con ácido clorhídrico (HCL) a reflujo para diferentes tiempos. Los resultados se presentan en la *Tabla 1.1* [27]. Se desea aplicar la **prueba t de Student** a la diferencia de las **medias** muestrales.

Tiempo de reflujo (min)	Estaño encontrado ($mg\,Kg^{-1}$)
30	55, 57, 59, 56, 56, 59
75	57, 55, 58, 59, 59, 59

Tabla 1.1: Resultados de la determinación de estaño

Solución tradicional:

Primero, se obtienen las **medias** para los dos tiempos:

$$M_x = \frac{1}{N}\sum_{i=1}^{N} D_{ix} = \frac{342}{6} = 57.00\ \ mg\,Kg^{-1} \tag{1.1}$$

$$M_y = \frac{1}{N}\sum_{i=1}^{N} D_{iy} = \frac{347}{6} = 57.83\ \ mg\,Kg^{-1} \tag{1.2}$$

Luego, se encuentra la **varianza** respectiva a cada tiempo:

$$V_x = \frac{1}{N-1}\sum_{i=1}^{N}(D_i - M_x)^2 = \frac{14.00}{5} = 2.80\ \ mg\,Kg^{-1} \tag{1.3}$$

$$V_y = \frac{1}{N-1}\sum_{i=1}^{N}(D_i - M_y)^2 = \frac{12.83}{5} = 2.57\ \ mg\,Kg^{-1} \tag{1.4}$$

A continuación, se calcula **el valor conjunto para la varianza**:

$$V_c = \frac{(N_x - 1)V_x + (N_y - 1)V_y}{(N_x + N_y) - 2} = \frac{(5 \times 2.80 + 5 \times 2.57)}{10} = 2.685\ \ mg\,Kg^{-1} \tag{1.5}$$

Finalmente, se obtiene el valor **t de Student**:

$$t = \frac{M_x - M_y}{\sqrt{\frac{V_c}{N_x} + \frac{V_c}{N_y}}} = \frac{57.00 - 57.83}{1.64\sqrt{\frac{1}{6} + \frac{1}{6}}} = -0.88 \tag{1.6}$$

Solución de acuerdo con el enfoque de *workflow*

Para este problema se crea el *workflow* que se muestra en la *Figura 1.6* y se guarda en la plataforma Taverna. Las entradas del *workflow* son los datos correspondientes a la determinación de estaño para los tiempos de reflujo que se muestran en la *Tabla 1.1*. A su salida se obtendrá el resultado de aplicar la prueba t de Student a la diferencia de las medias muestrales. Este resultado deberá corresponder al que se obtuvo mediante la solución tradicional.

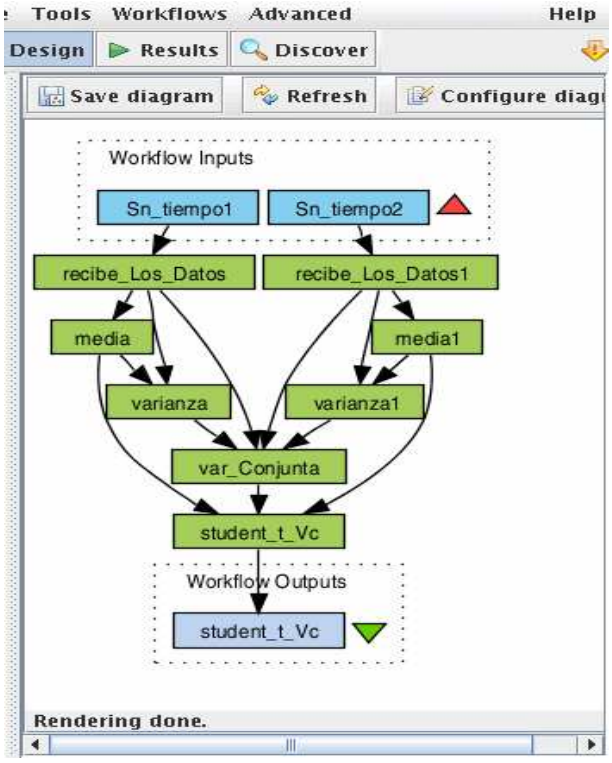


Figura 1.6: *Workflow* para la prueba t de Student

En el siguiente diagrama (*Figura 1.7*) se presentan las etapas por las que atraviesa el flujo de datos en un *workflow*. El diagrama detalla los estados de invocación de servicios, así como la eventualidad de abortar la invocación. En la etapa (I) se verifica que los datos existan a la salida del componente de software que alimenta al *workflow*. Si los datos están en el estado listos, entonces, se pasa a comprobar que éstos sean del mismo tipo que los requeridos por el servicio de la etapa siguiente. Por lo que los datos pasan al bucle señalado con la letra **A** en el diagrama. Si los datos son del mismo tipo, se invoca al siguiente servicio. En caso de que se presente algún error en la invocación o el número de intentos para invocar el servicio se agote (timeout) se abortará la ejecución del *workflow*. En caso contrario, la invocación será completada y el servicio regresará un resultado (en el punto señalado con la letra **B**) que se utilizará para alimentar la siguiente etapa.

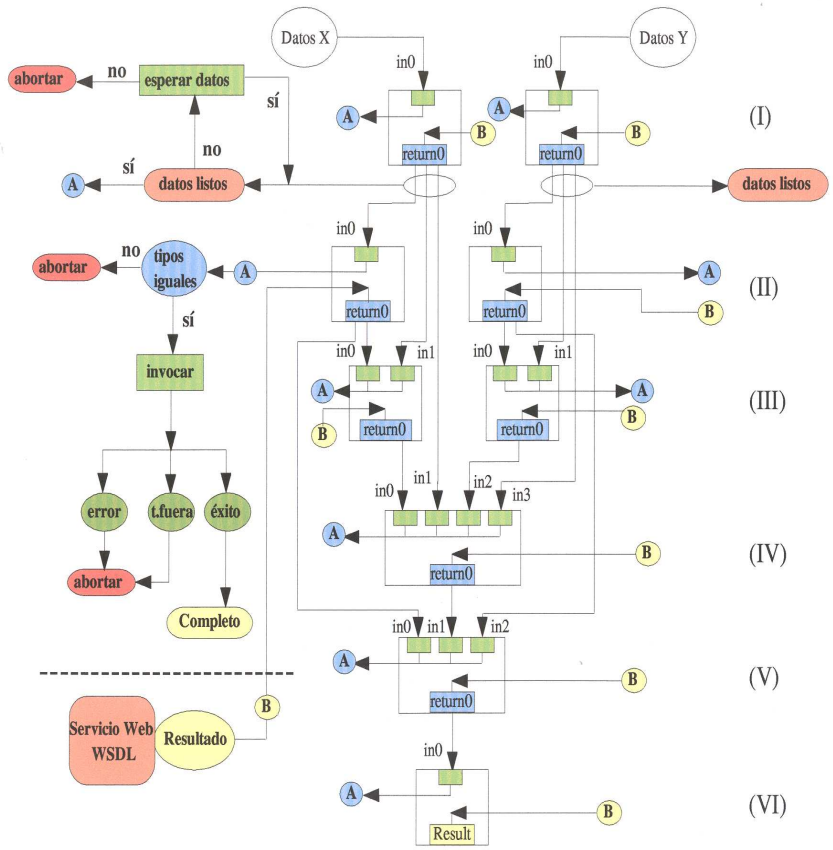


Figura 1.7: Diagrama con los estados para la invocación de un servicio

Este mismo proceso de verificación continuará para cada componente en todas las demás etapas hasta obtenerse el resultado final del *workflow* completo.

Para construir un *workflow* el usuario debe referirse al Apéndice A conteniendo las descripciones de los íconos representativos de todos los métodos implementados.

En el problema anterior, los componentes de software distribuidos que forman el *workflow* pueden provenir de diferentes sitios ubicados en el ciber espacio, permitiendo compartición de recursos tanto de hardware como de software. Ver *Figura 1.8*.

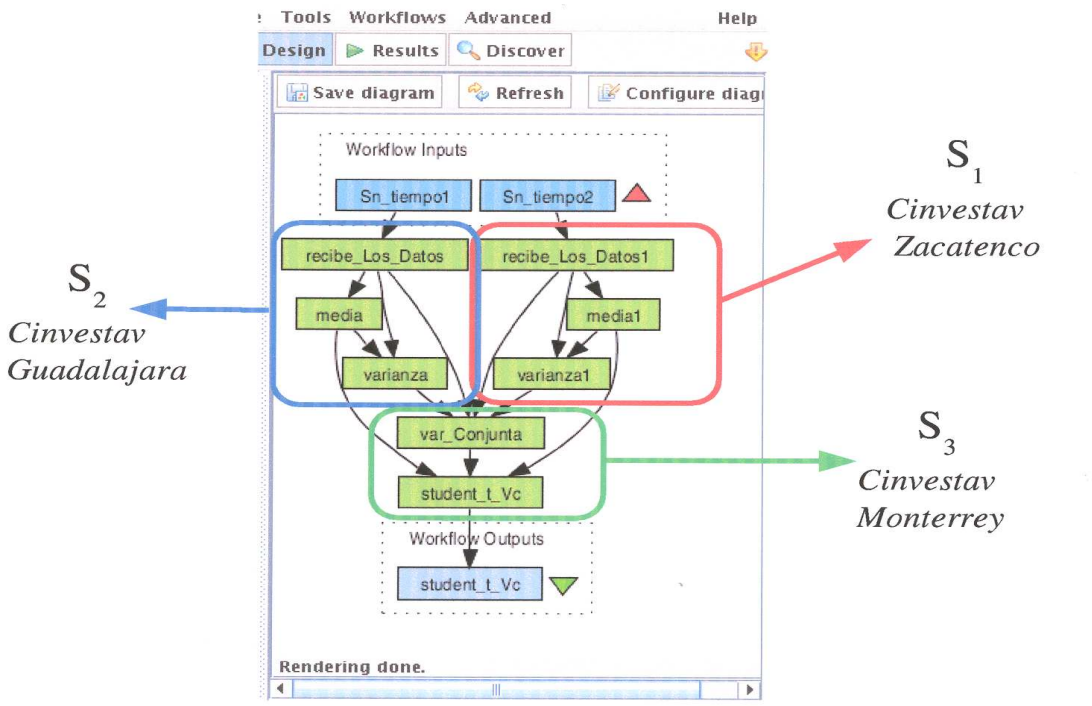


Figura 1.8: Los servicios pueden ser invocados desde diferentes sitios

6 Organización de la tesis

En el Capítulo 2 describimos las principales tecnologías o paradigmas en los cuales se basan los ambientes gráficos de *workflows*: servicios Web, *workflows* científicos y la arquitectura de capas [38]. Se introducen las plataformas gráficas más sobresalientes para *workflows* científicos actualmente en pleno desarrollo: Triana [77], Ptolomy [81], Kepler [78], SciRUN [79] y Taverna [41]. La última parte del Capítulo 2 se dedica a conocer los detalles más importantes de la plataforma Taverna relevantes a nuestra implementación.

En el Capitulo 3 describimos Tavlab, nuestro entorno para análisis estadístico de datos basado en *workflows* con Taverna.

En el Capítulo 4 explicamos cómo utilizar Tavlab para llevar a cabo el análisis de varianza, y aplicar las pruebas de Tukey y de Fisher a datos experimentales en el campo de la Biotecnología.

En el Capítulo 5 explicamos cómo utilizar Tavlab para realizar el analisis estadístico de datos experimentales en los campos de la Quimiometría y la Medicina Estadística. El grado de dificultad de los problemas propuestos aumenta gradualmente con la finalidad de hacer más intuitiva la utilización de nuestro entorno Tavlab.

En el Capítulo 6 exponemos las conclusiones y el trabajo futuro de este trabajo de investigación.

Capítulo 2

Taverna

Taverna es un ambiente gráfico distribuido de resolución de problemas que permite construir *workflows*[1] de procesos complejos mediante métodos ofrecidos por servicios Web[1]. Taverna se concibió en un principio para facilitar el uso e integración del creciente número de bases de datos y herramientas bioinformáticas en la Web[2]. Recientemente, investigadores de otras disciplinas han encontrado en Taverna una herramienta gráfica atractiva en la cual desarrollar aplicaciones. Tal es el caso de la Quimioinformática, la Imagenología Médica y los Sistemas Biológicos[1][2].

La primera versión de Taverna apareció disponible en el mes de febrero de 2003 como parte del proyecto *myGrid*[2] orientado a satisfacer los requerimientos de la comunidad científica del Reino Unido, con la participación del Instituto Europeo de Bioinformática (EMBL-EBI) y de varias universidades inglesas [3]. El proyecto *myGrid* surge a finales del año 2001 dentro de la *e-Science* para dar soporte a las diferentes plataformas de cómputo distribuido en las cuales se realizan diversas aplicaciones científicas, como el diseño y ejecución de *workflows*, así como la administración de datos y metadatos. El término *e-Science* se introduce en el año de 1999 en Inglaterra para describir procesos científicos demandantes de computación intensiva, la cual es llevada a cabo a través de redes altamente distribuidas.

En la *Sección 2.1* se introducen las características de los ambientes gráficos de *workflows*, en la *Sección 2.2* se explica cómo crear, ejecutar y guardar un *workflow* en Taverna, en la *Sección 2.3* se explica cómo realizar búsquedas semánticas de servicios Web con Feta [4][5] y en la *Sección 2.4* se introduce PeDRo [6], herramienta para captura y publicación de servicios Web.

[1] Término traducido frecuentemente en la literatura informática como diagrama de flujo o flujo de trabajo

[2] http://www.ebi.ac.uk/collab/mygrid

2.1 Ambientes gráficos de *workflows*

En esta sección describimos las principales tecnologías o paradigmas en los cuales se basan los ambientes gráficos de *workflows*: servicios Web, *workflows* científicos y la arquitectura de capas [38]. Así también, se introducen las plataformas gráficas más sobresalientes para *workflows* científicos actualmente en pleno desarrollo: Triana [77], Ptolomy [81], Kepler [78], SciRUN [79] y Taverna [41].

2.1.1 Servicios Web

Dada la popularidad adquirida por Internet en las últimas décadas, fueron desarrollándose intentos para permitir que sistemas heterogéneos pudieran mantenerse conectados a la Web intercambiando información.

El inconveniente surgió cuando las compañias implementaron sus propios sistemas de comunicación, muchas veces incompatibles con los de sus competidoras. La necesidad de integrar y compartir información entre distintas plataformas (tanto de software como de hardware) no sólo no se solucionó, sino que fue tomando cada vez una mayor dimensión.

Luego de un tiempo se llegó a la conclusión de que, en lugar de unificar los sistemas de intercambio de información, lo necesario era disponer de un lenguaje común para tal intercambio que respetara reglas estrictas y cuyo estándar estuviera definido. Así surge el lenguaje de marcado extensible XML (*eXtensive Markup Language*), un estándar para el intercambio de información estructurada entre diferentes plataformas, permitiéndonos tener acceso a datos en cualquer parte de la Internet. La adopción de XML como un estándar aplicado al intercambio de mensajes dio lugar al nacimiento de los servicios Web.

Un servicio Web es una colección de protocolos y estándares que sirven para intercambiar datos entre aplicaciones, considerándose así una plataforma integradora de aplicaciones [21]. De manera más precisa, un servicio Web es una manera estandarizada de integrar aplicaciones Web utilizando los estándares abiertos: lenguaje de marcado extensible XML (*eXtensive Markup Language*), protocolo simple de acceso a objetos SOAP (*Simple Object Access Protocol*), lenguaje de descripción de servicios Web WSDL (*Web Services Description Language*) y el protocolo de descripción, descubrimiento e integración Universales UDDI *(Universal Description, Discovery and Integration*) sobre el protocolo de transporte de hipertexto HTTP (HyperText Transport Protocol) de Internet [22].

La *Figura 2.1* describe la interacción de cada uno de estos estándares.

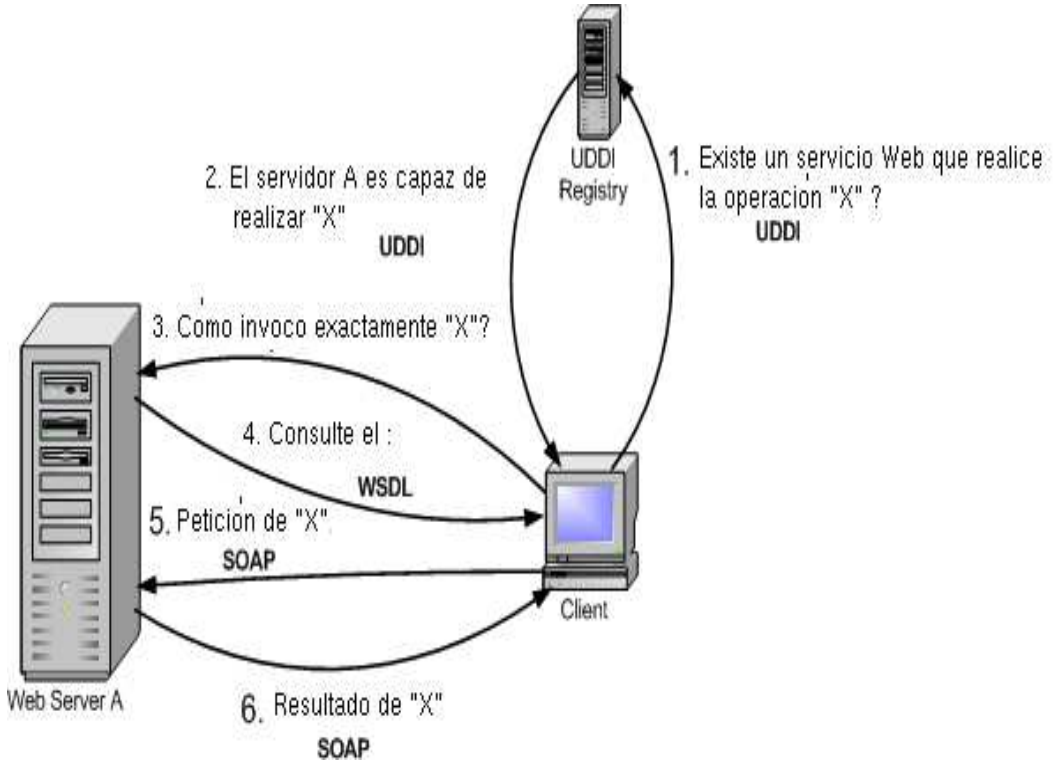


Figura 2.1: Interacción de los estándares de un servicio Web

1. El servidor UDDI es un registro o directorio de servicios Web. Una aplicación cliente pregunta al servidor UDDI por un servicio Web que ofrezca un método capaz de realizar la operación "X".

2. Si el servidor UDDI tiene registrado tal servicio Web, contesta con la URL del servidor Web (Web Server A) conteniendo al servicio Web.

3. El cliente contacta al servidor Web (Web Server A) y pregunta cómo invocar al método dentro del servicio Web que es capaz de realizar la operación "X".

4. El servidor Web envía el documento WSDL, el cual describe los parámetros de entrada y de salida del método.

5. El cliente envía un mensaje SOAP con los parámetros de entrada que requiere el método.

6. El método invocado envía el resultado de la operación "X" en un mensaje SOAP.

En el Capítulo 3 (Sección 3.3.2), se explican los detalles de estos protocolos relevantes a la implementación de nuestros servicios Web para aplicaciones estadísticas.

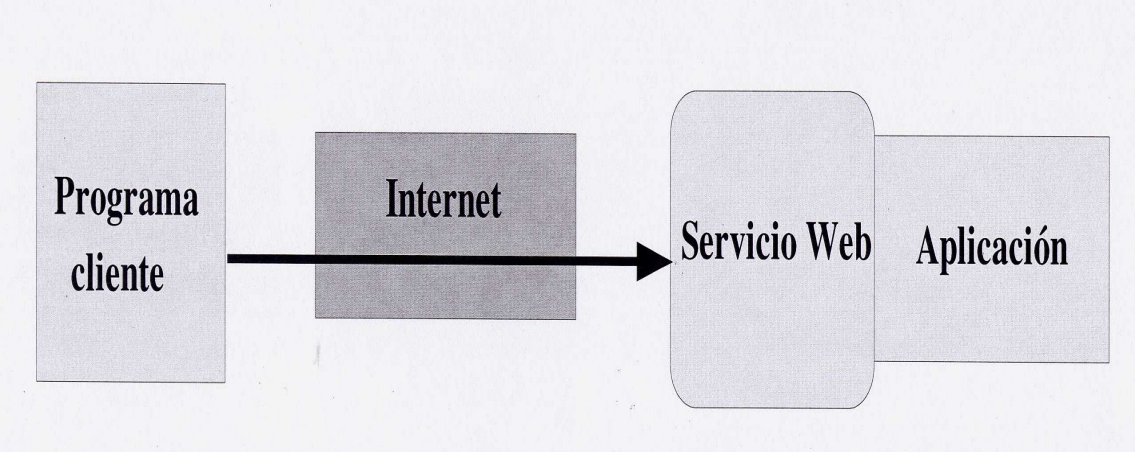


Figura 2.2: Solicitud de una aplicación ofrecida por un servicio Web

De esta manera, distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios Web para intercambiar datos en redes de computadoras como Internet.

Las organizaciones OASIS[3] y W3C[4] son los comités responsables de la arquitectura y reglamentación de los servicios Web. La *Figura 2.2* muestra un programa cliente solicitando una aplicación remota ofrecida por un servicio Web a través de Internet.

2.1.2 *Workflows* científicos

El concepto de *workflow* surge con la automatización del trabajo (work) en los procesos de producción y con la introducción de las líneas de ensamble (flow) hacia inicios del año 1900 [39]. De esta manera, la administración de *workflows* (o flujos de trabajo) originalmente fue considerada una disciplina confinada a la automatización de los procesos de negocios.

Actualmente, la mayoría de los los procesos de negocios dependen de la Internet y la administración de *workflows* evoluciona hacia una disciplina centrada en redes de ordenadores. Con esto, el alcance de la administración de *workflows* se amplía enormemente. Las ideas básicas y las tecnologías para la automatización de los procesos de negocios se extienden a virtualmente todas las áreas del quehacer humano, desde la ciencia e ingeniería hasta el entretenimiento. La coordinación de procesos provee los medios para mejorar la calidad del servicio, incrementar flexibilidad, permitir más alternativas y desarrollar servicios más complejos ofrecidos por proveedores independientes en la Web [38]. La alianza para la administración de *workflows* WfMC[5] (*Workflow Management Coalition*), es la organización mundial de usua-

[3]http://www.oasis.org
[4]http://www.w3c.org
[5]http://www.wfmc.org

rios, desarrolladores, consultores y analistas, así como de universidades y grupos de investigación relacionados con los sistemas de *workflows* y la administración de procesos de negocios. El WfMC crea estándares y contribuye en la capacitación de empresas que utilizan sistemas de *workflow* en la administración de sus procesos de negocio.

En términos informáticos, un *workflow* es un conjunto organizado y coordinado de componentes de software describiendo un proceso reproducible [19]. Un componente de software es un programa diseñado para realizar una tarea o aplicación específica dentro del proceso. Para poderse ejecutar, los componentes que forman el *workflow* necesitan de datos de entrada, así como de la verificación de precondiciones relacionadas con sus propias entradas. Definimos un componente de software como sigue [19]:

$$\begin{aligned} Componente\ de\ software ::= (&\langle Entrada\rangle, \langle Salida\rangle \\ &\langle Condición\rangle, \langle Resultado\rangle \\ &Nombre, Descripción) \end{aligned}$$

La *Figura 2.3* muestra a un componente de software definido por la Entrada A (5), la Entrada C (3), una Salida, un Nombre (*sumarAyC*), un Resultado (8), una Descripción (realiza la suma de 2 números) y una Condición (no ejecutar el componente B hasta que ambos componetes A y C hayan terminado).

La coordinación de la ejecucion de los diferentes componentes del *workflow* puede ser expresada a través de diferentes operadores:

$$\begin{aligned} Workflow ::= (&Componente | Workflow + Workflow | \\ &Workflow - Workflow | \\ &Workflow * | \\ &Workflow / Workflow) \end{aligned}$$

Esta definición establece que un *workflow* puede estar formado por un sólo componente o por una combinación de *workflows* estructurados por operadores de control. Estos operadores de control especifican que los componentes pueden ser ejecutados concurrentemente (operador +) o secuencialmente (operador -), que ejecutan iterativamente un sólo componente (operador *) o ejecutan alternativamente diferentes tareas (operador /). La construcción de un *workflow* mediante estos operadores define un flujo de control.

La formalización propuesta por Marinescu [38] para el problema de la coordinación de un sistema de *workflow* establece que dado:

(i) Un sistema $\sum$, un estado inicial del sistema, $\sigma_{inicial}$ y un estado final, σ_{final}.

(ii) Un grupo de procesos, $\boldsymbol{P} = \{p_1, p_2, ...p_n\}$. Cada proceso p_i, en el grupo de

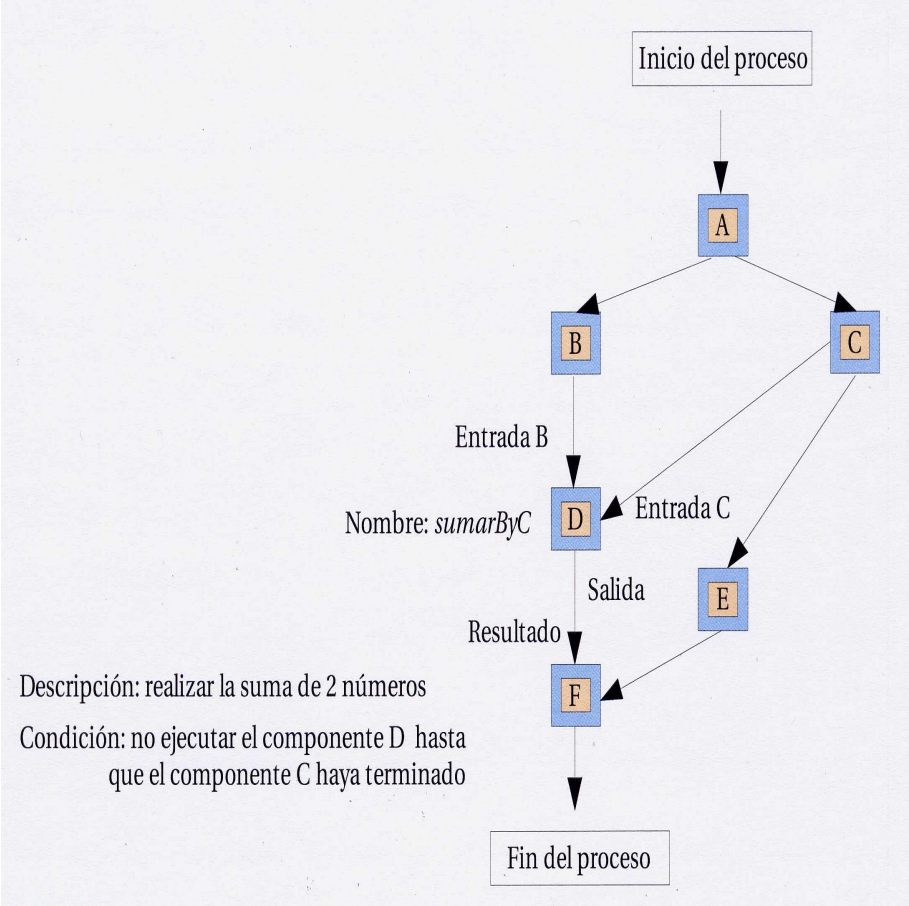


Figura 2.3: Componente de software dentro de un *workflow*

procesos, está caracterizado por un conjunto de precondiciones, $pre(p_i)$, postcondiciones, $post(p_i)$, y atributos, $atr(p_i)$.

(iii) Un *workflow* descrito por un grafo dirigido $\boldsymbol{A}$ o por un procedimiento $\prod$ capaz de construir $\boldsymbol{A}$ dado la tupla $\langle \boldsymbol{P}, \sigma_{inicial}, \sigma_{final} \rangle$. Los nodos de $\boldsymbol{A}$ son procesados en $\boldsymbol{P}$ y las aristas definen relaciones de referencia entre processos. $P_i \rightarrow P_j$ implica que $pre(p_j) \subset post(p_i)$.

(iv) Un conjunto de restricciones $\boldsymbol{C} = \{C_1, C_2, ...C_m\}$.

El problema de coordinación para el sistema $\sum$ radica en alcanzar el estado σ_{final} a partir del estado $\sigma_{inicial}$, como resultado de las postcondiciones de algún proceso $P_{final} \epsilon \ \boldsymbol{P}$ sujeto a las restricciones $C_i \epsilon \ \boldsymbol{C}$. Aquí, $\sigma_{inicial}$ da lugar a las precondiciones de algún proceso $P_{inicial} \epsilon \ \boldsymbol{P}$.

Generalmente, las precondiciones de un proceso o los datos que el proceso espera como entrada, constituyen las condiciones que disparan su ejecución y las postcon-

diciones son el resultado producido por el proceso. Los atributos de un proceso describen procedimientos especiales o propiedades del proceso.

Los sistemas de *workflows* científicos se caracterizan principalmente por manejar flujos de datos experimentales. Los componentes de software de los *workflow* científicos son primordialmente servicios Web de aplicaciones remotas distribuidas que realizan cálculos complejos. La *Figura 2.4* muestra un *workflow* científico construido para una aplicación en Bioinformática.

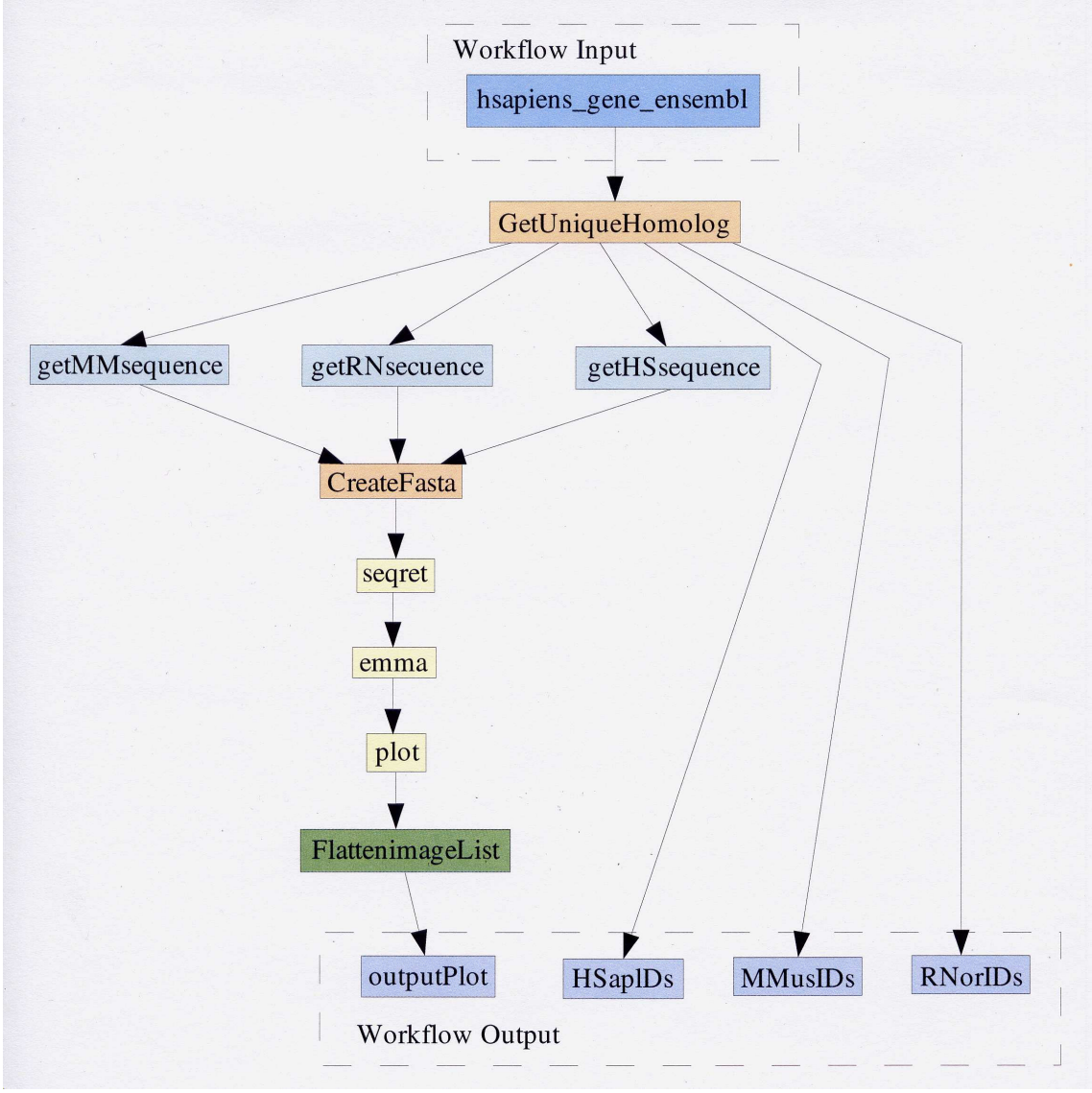


Figura 2.4: *Workflow* científico en Bioinformática

2.1.3 Arquitectura de capas

La arquitectura de capas consiste en dividir la implementación de un ambiente de *workflows* en 3 módulos principales e independientes, de forma que cada uno de ellos tenga funcionalidades propias. Esto permite a las capas, en determinado momento, ser sustituidas por una implementación nueva sin afectar a las capas adyacentes. La *Figura 2.5* muestra las capas que intervienen en el funcionamiento de un *workflow*.

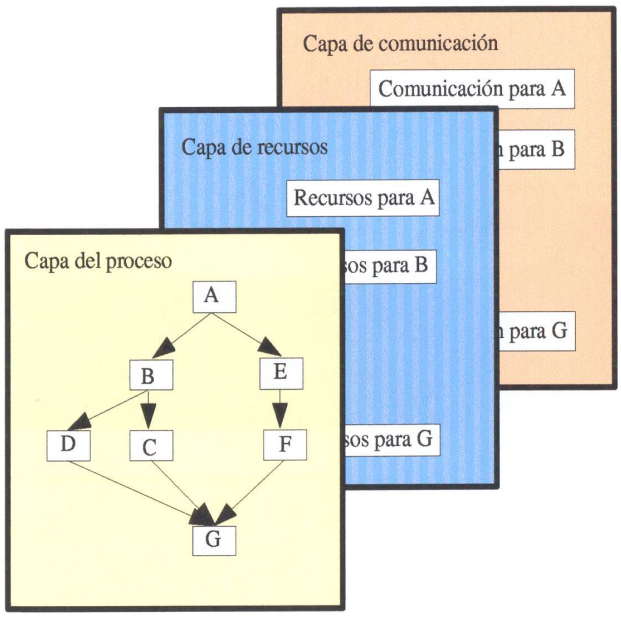


Figura 2.5: Arquitectura de capas

1. Capa del proceso: consiste de la interfaz gráfica del sistema y permite interactuar con las diferentes representaciones de componentes de software, propias del sistema, para formar un *workflow*.

2. Capa de localización de recursos: en esta capa se encuentran las líneas de código XML o derivado de XML (como el lenguaje SCUFL[6]) describiendo el *workflow* y con las direcciones URL donde localizar a los diferentes recursos de servicios Web durante su ejecución.

[6]Simple Conceptual Unified Flow language, utilizado por la plataforma Taverna

3. La capa de comunicación: en esta capa se realizan actividades tales como el traslado de los datos del consumidor al productor (y viceversa), comprimir y descomprimir datos, encriptación y decodificación de datos, así como transformaciones de formatos.

2.1.4 Plataformas gráficas para *workflows* científicos

A continuación, se intruducen las plataformas gráficas más sobresalientes para *workflows* científicos actualmente en pleno desarrollo: Triana [77], Ptolomy [81], Kepler [78], SciRUN [79] y Taverna [41].

I) **Triana**: es un proyecto de la Universidad de Cardiff en Inglaterra enfocado a realizar tareas de procesamiento de señales, de texto y de imágenes. La interfaz de usuario de Triana consiste de un conjunto de módulos (toolboxes) conteniendo componentes pre-cargados al momento de su instalación para realizar procesamientos de señales de audio, imágenes y texto. Además, Triana cuenta con componentes para las instrucciones principales de programación como son aquellas para controlar ciclos de procesamiento: *do*, *while*, *repeat until*, etc. y para acciones lógicas: *if then* , etc. Esto permite controlar el flujo de datos gráficamente, precisamente como un programador lo haría con instrucciones de la programación convencional. Las gráficas de los *workflows* en Triana se construyen en formato XML y cuentan con la funcionalidad para interectuar con servicios Web. El sitio oficial del proyecto Triana es: http:// trianacode.org. La *Figura 2.6* muestra la interfaz de usuario de Triana.

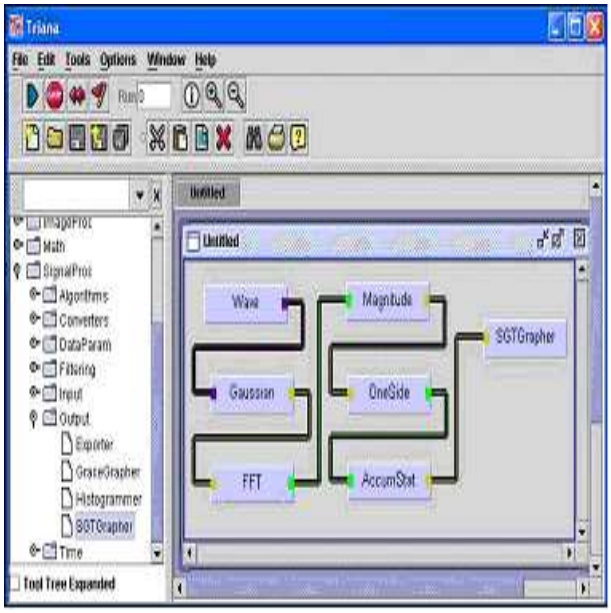


Figura 2.6: Interfaz de usuario de Triana

II) **Ptolemy**: es un proyecto de la Universidad de California en los E.U.A. enfocado al modelado, simulación y diseño de sistemas concurrentes, de tiempo real y embebidos. En Ptolemy a cada componente que forma un *workflow* se le llama *actor*. Un *actor* puede tener puertos múltiples de entrada y salida, a través de los cuales fluyen los datos. Adicionalmente, los *actors* cuentan con parámetros para definir sus comportamientos específicos dentro de un *workflow*. El sitio oficial del proyecto Ptolemy es: http://ptolemy.eeecs.berkeley.edu. La *Figura 2.7* muestra la interfaz de usuario de Ptolemy.

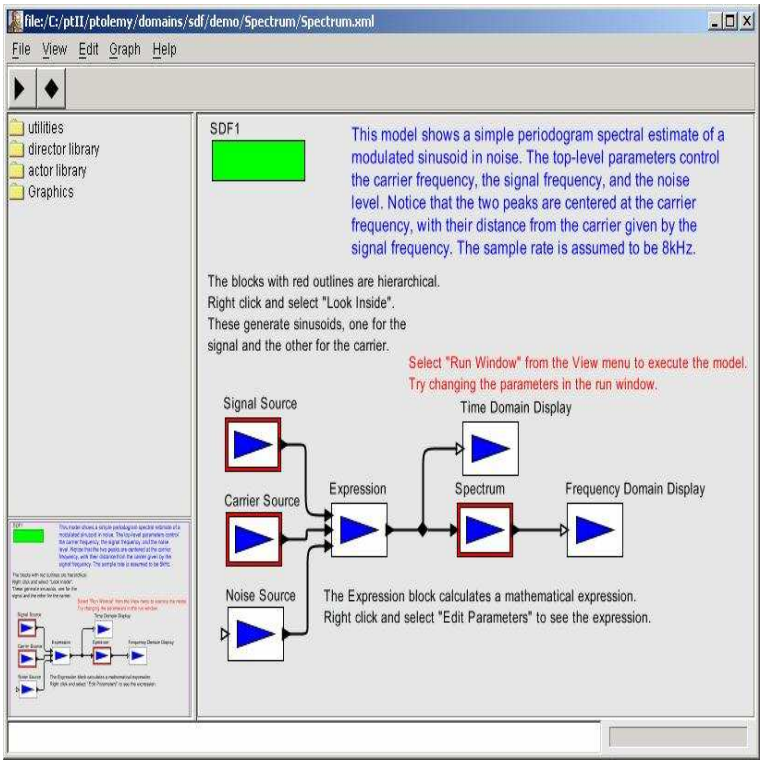


Figura 2.7: Interfaz de usuario de Ptolemy

III) **Kepler**: es un proyecto de la Universidad de California en los E.U.A. enfocado a las áreas de la Astrofísica, la Ecología y la Geología. Kepler es el proyecto hermano de Ptolemy heredando características y términos de éste. Sin embargo, las funcionalidades de Kepler se han extendido para aprovechar recursos computacionales, tales como los servicios Web. Por ejemplo, cuenta con un *WebService actor* para facilitarle al usuario la conexión y ejecución de métodos pertenecientes a servicios Web definidos en un documento WSDL. El sitio oficial de Kepler es: http://kepler-project.org. La *Figura 2.8* muestra la interfaz de usuario de Kepler.

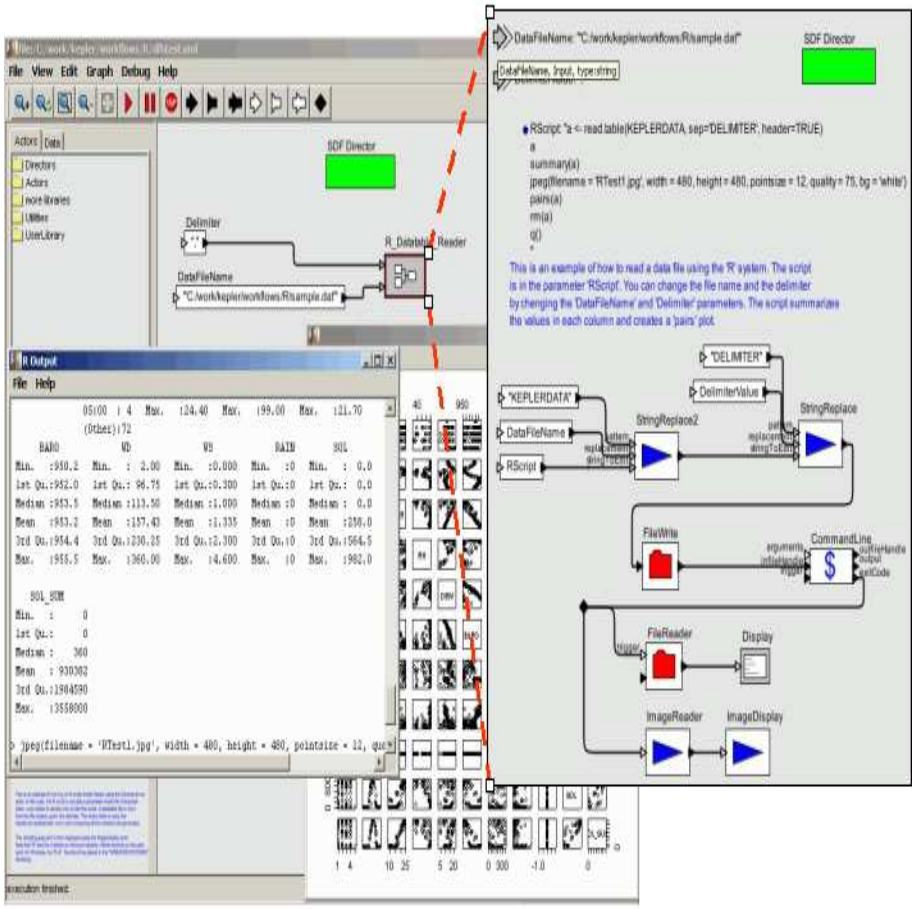


Figura 2.8: Interfaz de usuario de Kepler

IV) **SciRUN**: es un proyecto de la Universidad de Utah en los E.U.A. para realizar análisis de imágenes biomédicas. SciRUN se diseñó para realizar simulaciones a gran escala y para visualizar los datos en diferentes etapas del análisis. El sitio oficial de SciRUN es: http://software.sci.utah.edu/scirun.html. La *Figura 2.9* muestra la interfaz de usuario de SciRUN.

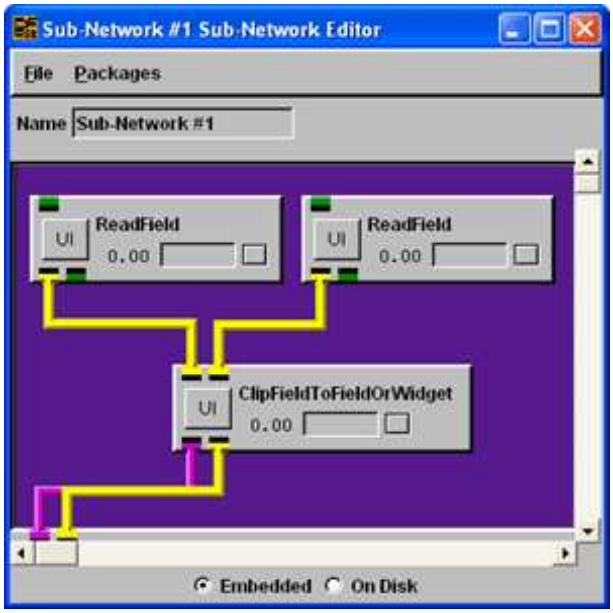


Figura 2.9: Interfaz de usuario de SciRUN

V) **Taverna**: forma parte del proyecto *myGrid*[2] desarrollado por el Instituto Europeo de Bioinformática y varias universidades inglesas para facilitar el uso y la integración del creciente número de bases de datos y herramientas bioinformáticas en la Web. Taverna es actualmente una plataforma bien establecida y ampliamente utilizada para llevar a cabo experimentos con *workflows* en la *e-Science* [1]. En Taverna todas las etapas computacionales de los *workflows* son servicios Web. Es importante destacar que OGSA-DAI[7] es el middleware a través del cual Taverna puede llevar a cabo experimentos *in silico*[8] a gran escala, predominantemente en las áreas de las ciencias de la vida. Este middleware también forma parte del proyecto *myGrid*. El sitio oficial de Taverna es: http://taverna.sourceforge.net. La *Figura 2.10* muestra la interfaz de usuario de Taverna.

[2]http://www.ebi.ac.uk/collab/mygrid

[7]OGSA-DAI facilita el acceso e integración de datos que provienen de fuentes distribuidas vía la Grid. http://www.ogsadai.org.uk

[8]Es una expresión que significa "realizado en computadora o una simulación empleando computadoras". Este término fue usado primero en público en el año 1989 por el Dr. Pedro Miramontes, matemático de la UNAM, en un taller de Autómatas Celulares realizado en Los Alamos, Nuevo México.

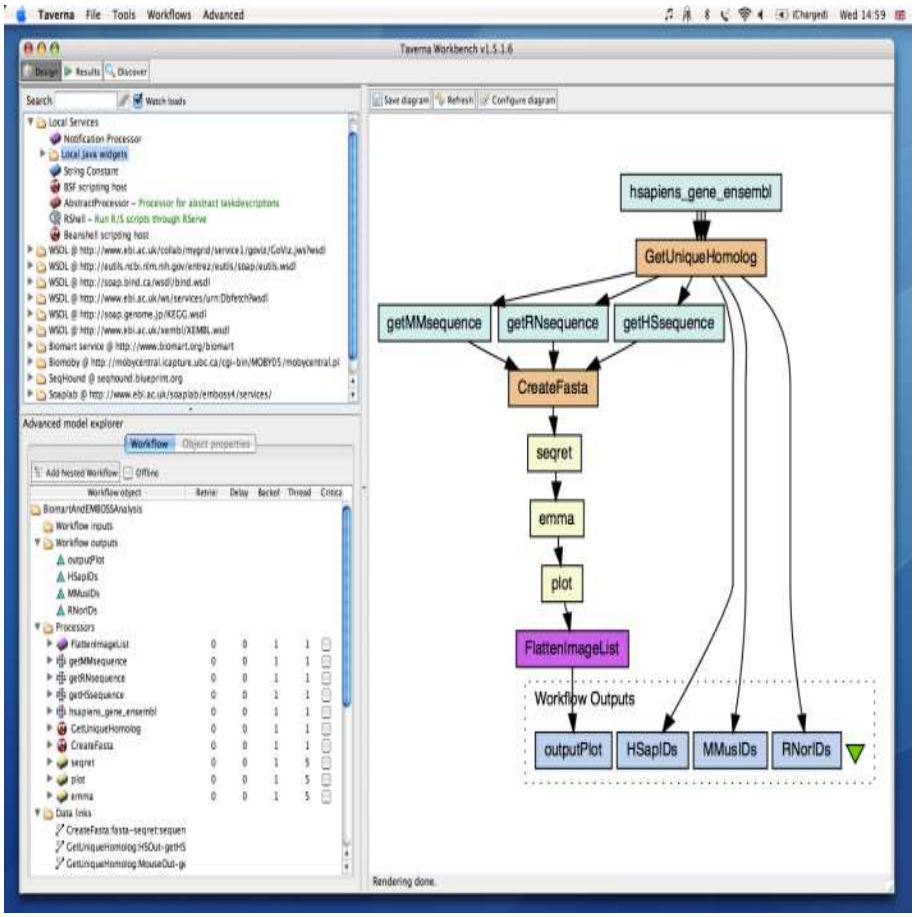


Figura 2.10: Interfaz de usuario de Taverna

Cabe mencionar que el proyecto Taverna ha ocupado un lugar importante en las dos primeras conferencias internacionales sobre *e-Science* y la computación *Grid* organizadas por la IEEE en diciembre del 2005 [1] y del 2006 [18]. Así también, ha sido tema de publicación en revistas arbitradas internacionales como: Practice and Experience [3], Bioinformatics [7] y Nucleic Acids Reseach [2].

En la siguiente sección nos centraremos en la plataforma Taverna y explicaremos cómo utilizarla.

2.2 Creación de un *workflow* en Taverna

En esta sección se presentan los pasos principales para construir un *workflow* en Taverna. Se realiza una aplicación idónea para iniciar la construcción de *workflows* en esta plataforma: un *workflow* denominado calculadora. Para esto, se ha creado el servicio Web *CalculadoraWS*. Los detalles de la instalación de Taverna se encuentran en el Apéndice A.

2.2.1 Pasos para crear un *workflow* en Taverna

Una vez que el software de Taverna se ha descargado, instalado y ejecutado, aparecen tres ventanas que corresponden a los **servicios disponibles** (Available services), al **explorador** (Advanced model explorer) y a la de los **diagramas** (Workflow diagram). En la *Figura 2.11* se muestran estas ventanas. Construir una aplicación en Taverna es sencillo siempre y cuando se tenga claro qué se desea analizar y cómo se desea hacer. En general, para usar Taverna deben seguirse los siguientes pasos, en los cuales las selecciones de las diferentes ventanas se llevan a cabo mediante la pulsación del botón derecho del *mouse*:

(1) Se debe seleccionar de la ventana de **servicios disponibles** el servicio Web que se desea utilizar. Para hacer esto, en la carpeta de **"Available Processors"** se selecciona la opción **"Add new WSDL Scavenger"**, y en la ventana localizadora que aparece se introduce la URL del servicio Web requerido. Finalmente, se selecciona **"OK"**.
(2) A continuación, aparecerá añadido el servicio Web en la misma ventana de **servicios disponibles**. En seguida, se expande el servicio Web para seleccionar los métodos que se requieran y se selecciona **"Add to model"** en el menú que aparece para cada método seleccionado.

(3) En la ventana de **diagramas** aparecerán estos métodos como bloques. Ahora se necesitan especificar las entradas y salidas. Esto se hace en la ventana del **explorador**. Se escoge la opción **"Workflow inputs"**, se selecciona **"add an input"** y se introduce un nombre para la entrada. Esto se hace para cada entrada que requiera la aplicación. De manera similar se crea la salida. Esto es, se elige **" Workflow outputs"**, se selecciona **"add an output"** y se introduce un nombre para la salida. Esto se hace para cada salida que requiera la aplicación. Se pueden ver las entradas y salidas que recién se crearon en la ventana del **explorador**, así como en bloques en la ventana de **diagramas**.

(4) Ahora simplemente se vinculan los métodos con las entradas y las salidas. Desde la ventana del **explorador** se seleccionan los métodos y en el menú que aparece se escoge la entrada y salida para cada uno de ellos. Se verá cómo el diagrama se irá construyendo cada vez que se forma un vínculo.

(5) Después se alimenta el proceso con los datos requeridos y se ejecuta. En el panel principal se selecciona **"File"** y en seguida **"Run Workflow"**. Aparecerá una ventana en la cual se deben introducir los datos. Esto se puede hacer a partir de un archivo o introduciendo los datos por el teclado. Cuando, se oprime el botón **"Run Workflow"** ocasiona que la aplicación se ejecute.

(6) Finalmente, aparecerá la ventana **"Enactor invocation"** con varias pestañas etiquetadas con los nombres de las salidas. Al seleccionar estas pestañas se mostrarán los resultados. Esta ventana también cuenta con las pestañas para el **"Status"** y el **"Process report"**, en las cuales se presentará el estado de ejecución del proceso y el reporte del proceso respectivamente. Esta información es muy útil en caso de algún error.

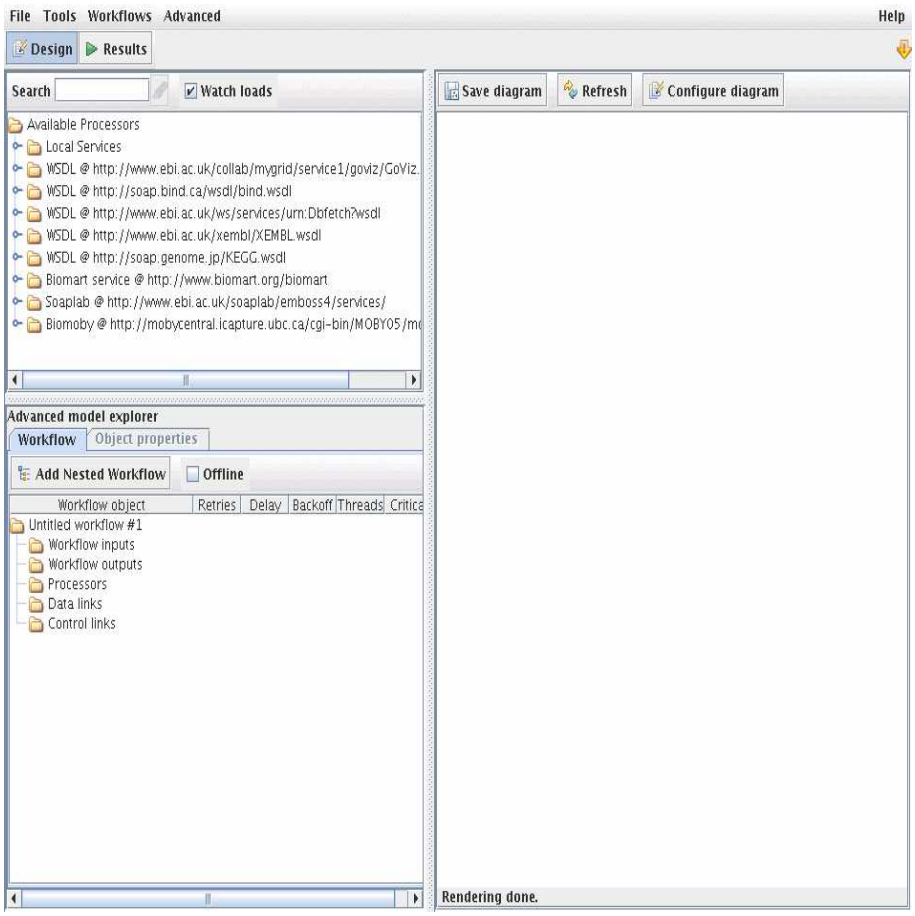


Figura 2.11: Interfaz de Taverna

Una característica interesante de Taverna es que una vez construido el *workflow* para algún proceso, permite guardarlo en el disco duro para poderlo reutilizar e incluso compartir. Igualmente, los resultados obtenidos pueden guardarse en un archivo para su análisis posterior[40].

2.2.2 Ejemplo de la calculadora

A partir del servicio Web *CalculadoraWS* se desea realizar el *workflow* que calcule la suma, resta, multiplicación y división de las entradas x e y. Siguiendo los pasos indicados en la sección anterior se tiene:

En la ventana de **servicios disponibles** se elige la opción **"Available Processors"** y en el menú que aparece se selecciona la opción **"Add new WSDL Scavenger"**. A continuación, se muestra una pequeña ventana localizadora en la cual se introduce la siguiente URL: **http://localhost:8080/axis/services/ CalculadoraWS?wsdl** y se oprime **"OK"**. Ver *Figura 2.12*.

Figura 2.12: Ventana localizadora en la cual se introduce la URL del Servicio Web

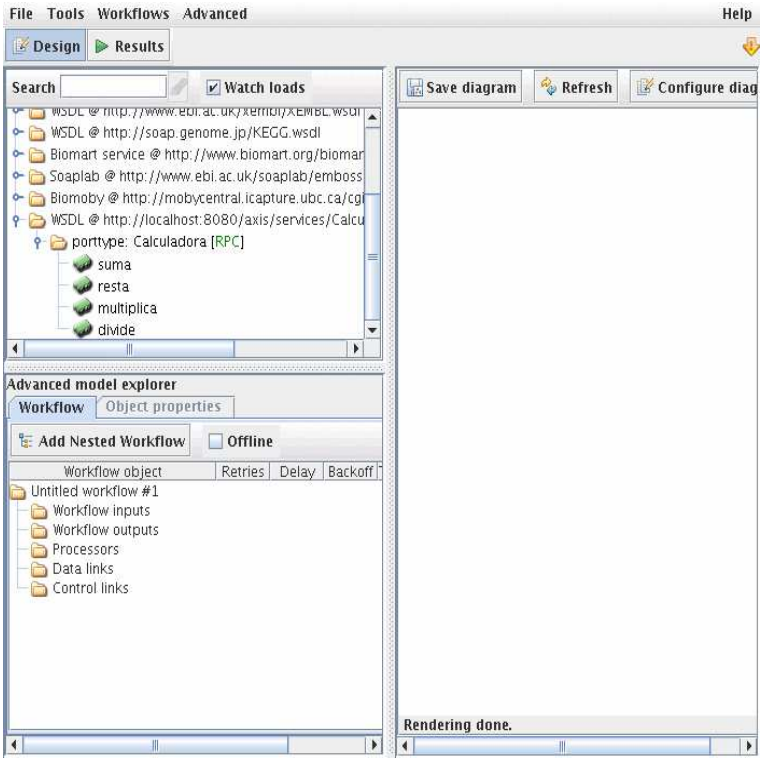


Figura 2.13: Servicio Web CalculadoraWS expandido

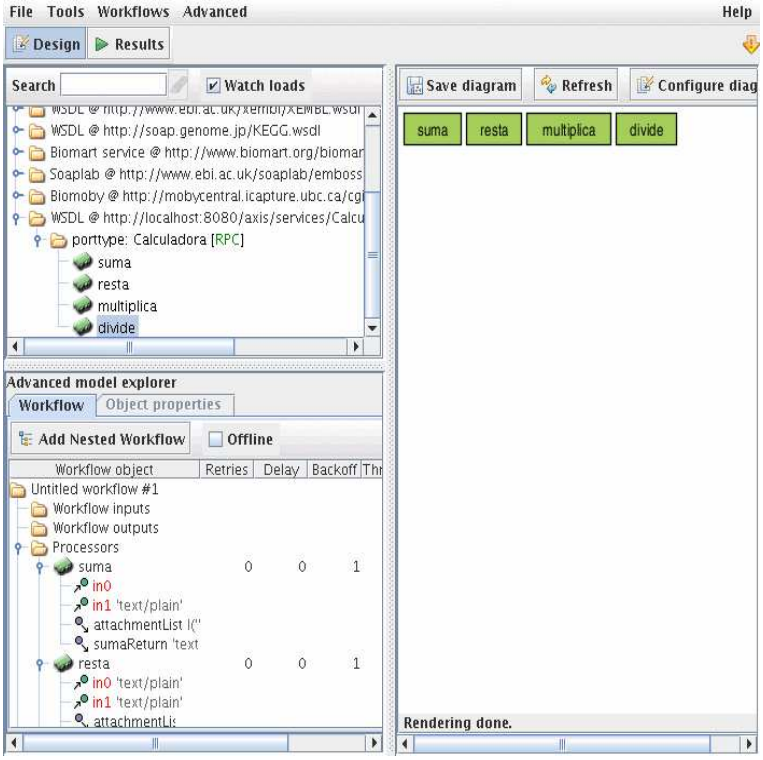


Figura 2.14: Métodos agregados a la ventana de diagramas

Después de esto, el servicio Web *CalculadoraWS* es añadido a la ventana de **servicios disponibles**. Al expandirlo, dando un *click* en su carpeta, aparecen los métodos para la suma, resta, multiplicación y división. Ver *Figura 2.13*. Cuando se selecciona un método aparece un menú en el cual se elige la opción **"Add to model"** para agregar el método a la ventana de **diagramas**. Cada método agregado de esta forma a la ventana de **diagramas** se simboliza mediante un bloque de color verde etiquetado con su nombre. Ver *Figura 2.14*.

A continuación, en la ventana del **explorador** se escoge **"Workflow inputs"**, se selecciona **"add an input"** y se introduce: **entradaX**. Nuevamente se escoge **"Workflow inputs"**, se selecciona **"add an input"** y se introduce ahora: **entradaY**. De manera similar, para las salidas se escoge **"Workflow outputs"**, se selecciona **"add an output"** y se introduce: **salida1**, para el resultado de la suma. Esto se repite para la **salida2**, **salida3** y **salida4** correspondientes a los resultados de la resta, multiplicación y división. Ver *Figura 2.15*.

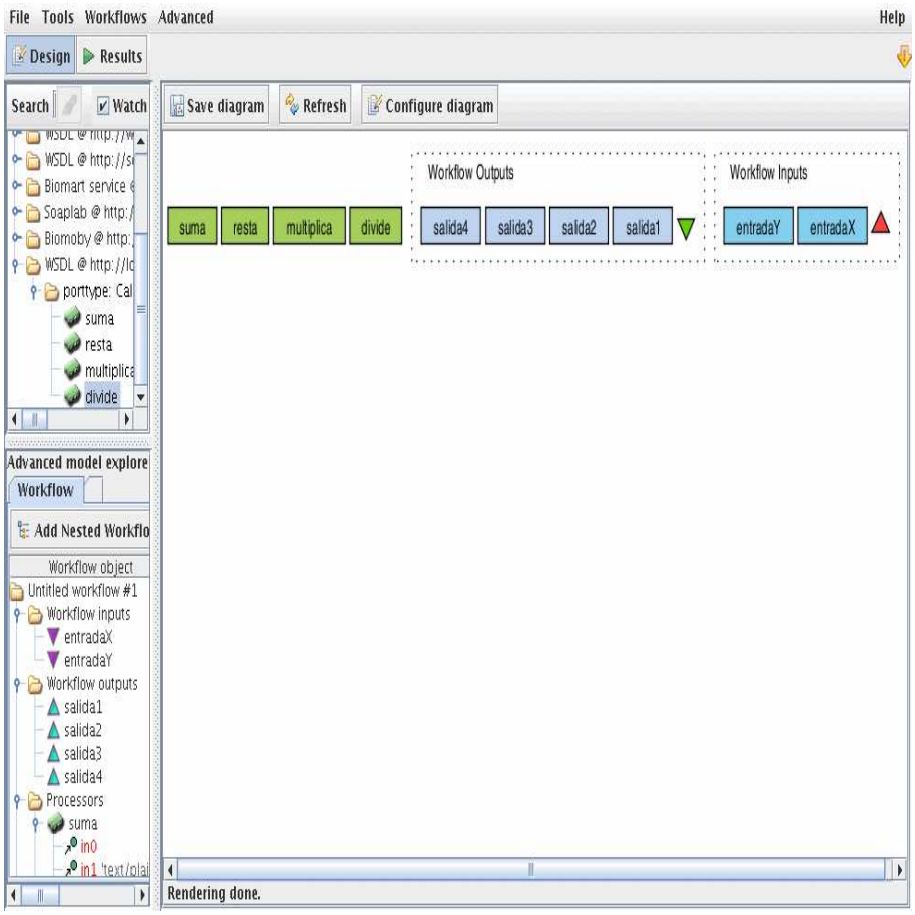


Figura 2.15: Entradas y salidas agregadas

Desde la ventana del **explorador**, bajo la carpeta **"Workflow inputs"**, se selecciona **entrada X** y se conecta su salida con las entradas de cada uno de los métodos. Luego, se selecciona **entrada Y** y se conecta su salida con las entradas de cada uno de los métodos. Asimismo, bajo la carpeta **"Processors"** se selecciona **suma** y se conecta su salida con la entrada de **salida1**. Esto se repite para los otros métodos. Como se puede apreciar en la *Figura 2.16* el *workflow* está listo para ejecutarse.

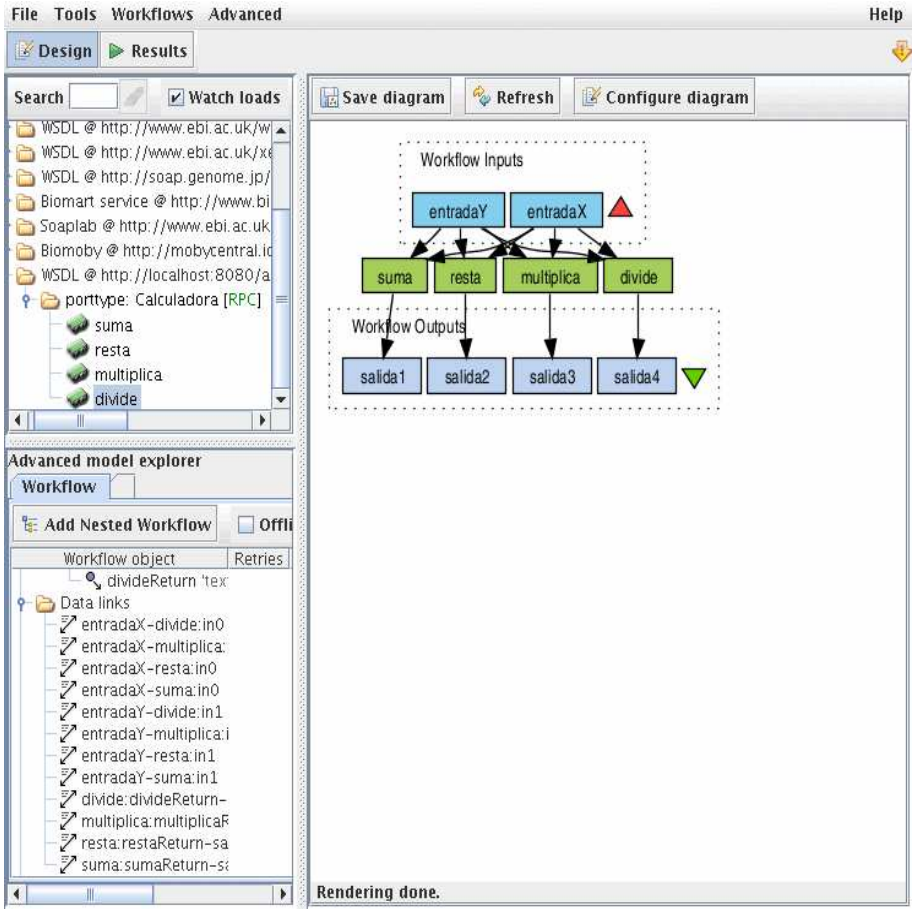


Figura 2.16: *Workflow* de la aplicación aritmética

2.2.3 Ejecución del *workflow*

Una vez creado el *workflow* aritmético, se ejecuta de la siguiente manera:

En el panel principal se selecciona **"File"** y enseguida **"Run Workflow"**. A continuación, aparece una ventana que muestra en su lado izquierdo las entradas tal y como se etiquetaron: **entradaX** y **entradaY**. En su parte superior hay un panel con botones para introducir datos. Se selecciona **entradaX** y se oprime el botón **"New Input"** con lo cual debajo de **entradaX** aparece un pequeño cursor para introducir el valor de la entrada. Se introduce 90 y se selecciona ahora la **entradaY** y de manera similar se le da el valor 45. En la parte inferior derecha de esta ventana se encuentra el botón **"Run Workflow"**, el cual se oprime para ejecutar el *workflow*. Esto se muestra en la *Figura 2.17*.

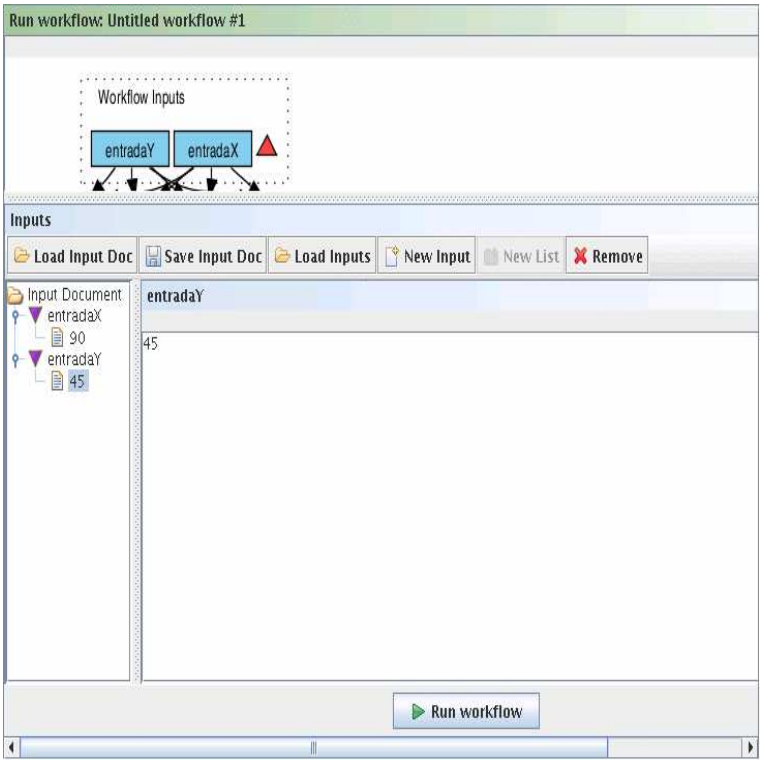


Figura 2.17: Ejecución del *workflow*

2.2.4 Revisión de resultados

Las pestañas de resultados

Unos cuantos segundos después de haber ejecutado el *workflow*, se presenta la ventana de resultados **"Enactor Invocation"** con las pestañas **"Status"**, **"Results"** y **"Process report"**. A su vez, la pestaña de **"Results"** incluye las pestañas con los nombres de las salidas: **salida1**, **salida2**, **salida3** y **salida4**. En cada una de estas salidas se encuentra el resultado de la suma, resta, multiplicación y división respectivamente. En la *Figura 2.18*, se muestra el resultado para la suma que corresponde a la pestaña **salida1**. Al seleccionar la pestaña **"Status"**, se muestra el estado de ejecución del proceso con los bloques que componen el *workflow* de color verde indicando que se ejecutaron exitosamente. Ver *Figura 2.19*. Cuando alguno de los nodos de un *workflow* no se ejecuta correctamente su estado de ejecución se mostrará de color rojo. Al pulsar ahora sobre la pestaña **"Process report"**, se muestra un reporte con la palabra *"COMPLETE"* de color verde en la parte superior . Esto se muestra en la *Figura 2.20*. Cuando hay un error, este reporte indica también en color rojo la etapa del proceso que no funcionó.

Figura 2.18: El resultado para la suma se encuentra en la pestaña **salida1**

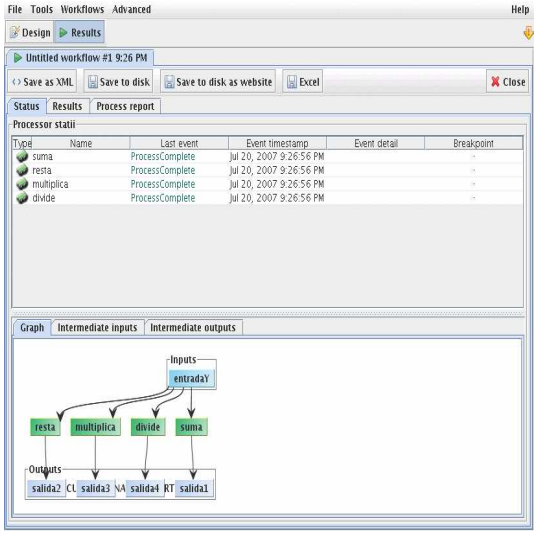


Figura 2.19: Ventana correspondiente a la pestaña **Status**

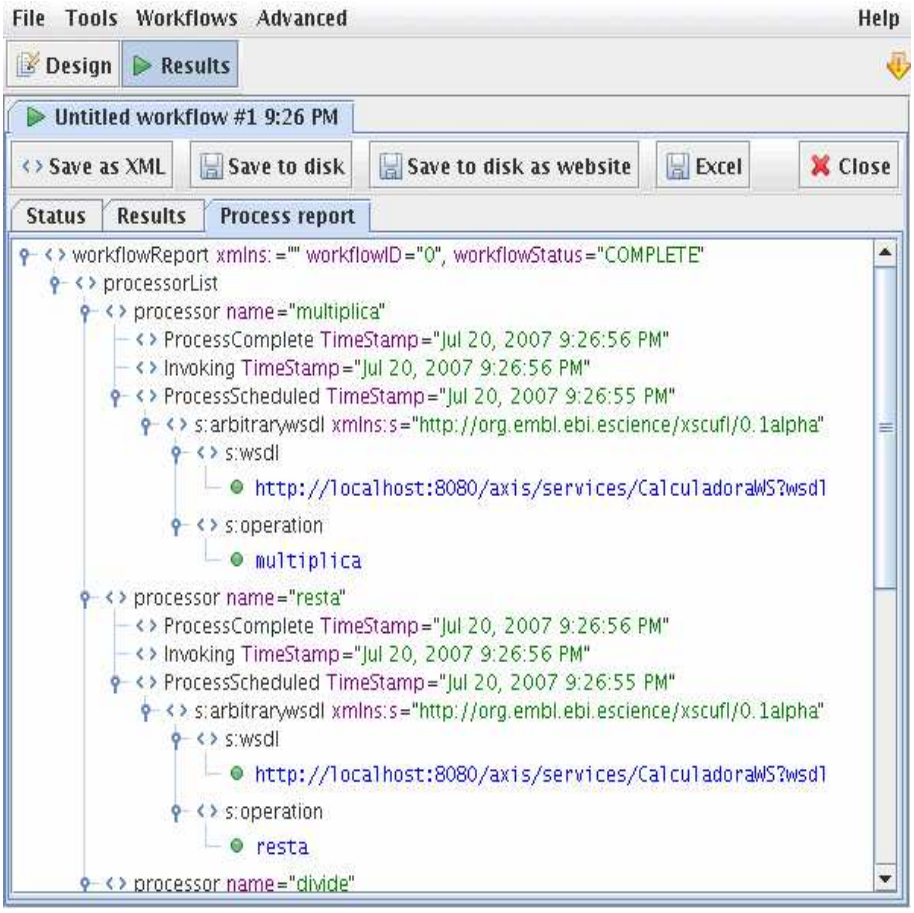


Figura 2.20: Ventana correspondiente a la pestaña **Process report**

Guardar los resultados

La ventana **"Enactor Invocation"** cuenta con varias opciones para guardar los resultados mostrados en cada una de las pestañas de salida. Los resultados pueden guardarse individualmente o en conjunto. Para guardarlos individualmente, se pulsa el botón derecho del *mouse* sobre el resultado y se selecciona **"Save to file"** en el menú que aparece. En seguida, se crea un archivo con mismo nombre de la salida y se guarda. Así, para el resultado de la **salida1**, el archivo que se crea tiene este mismo nombre. Ver la *Figura 2.21*. Para guardarlos como un conjunto de resultados, se oprime el botón **"Save to disk"** y a continuación se pide introducir el nombre del nuevo directorio dentro del cual todos los resultados del *workflow* serán almacenados. En la *Figura 2.22*, se ha creado el directorio **Resultados** y en la *Figura 2.23*, se muestran los archivos en que Taverna guardó los resultados dentro de este directorio. Las otras opciones para guardar resultados son: **"Save as XML"**, **"Save to disk as website"** y **"Excel"**.

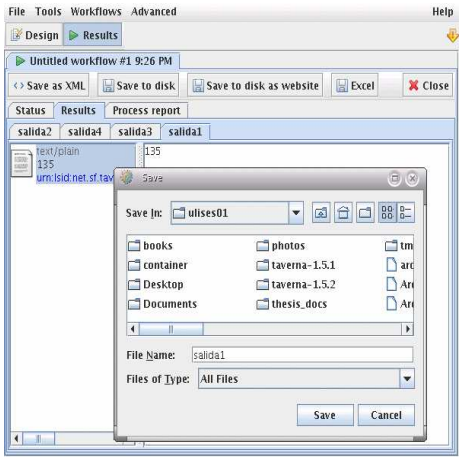


Figura 2.21: Se crea el archivo **salida1**

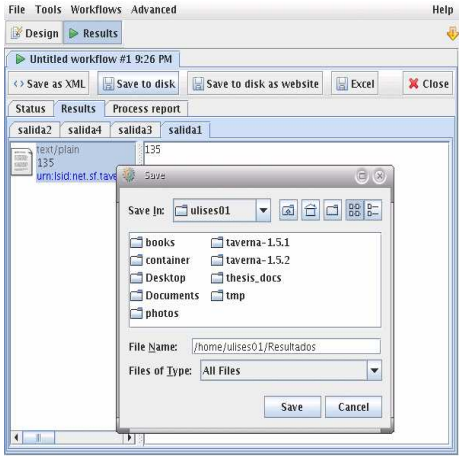


Figura 2.22: Se crea el directorio **Resultados**

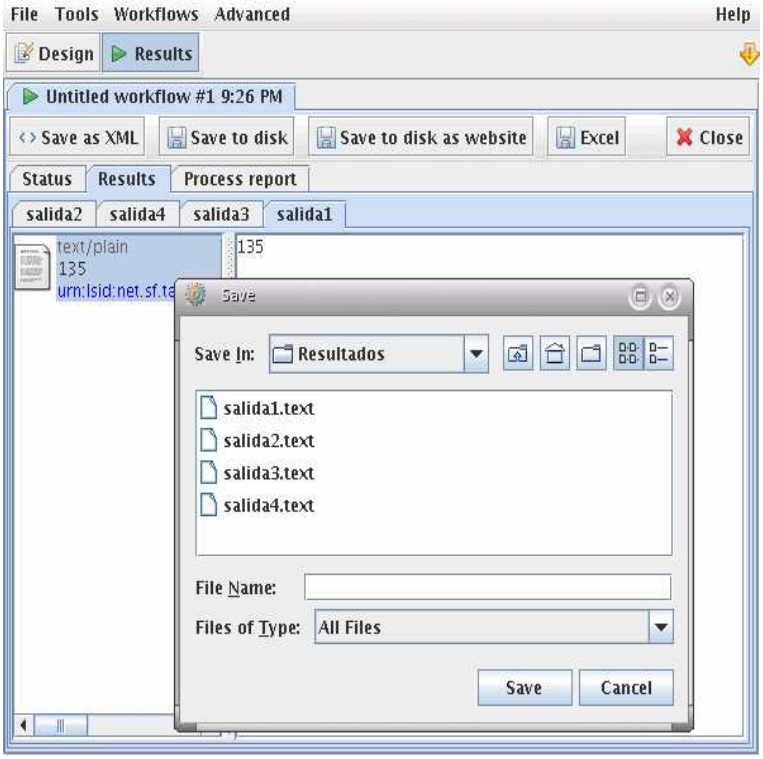


Figura 2.23: Archivos en que Taverna guardó los resultados

2.3 Búsqueda semántica de servicios con Feta

Feta es el componente dentro del projecto *myGrid* responsable de la búsqueda semántica de servicios Web[6]. Feta está compuesto de dos elementos, el *Feta Client* y el *Feta Engine*. El *Feta Client* es una interfaz gráfica de usuario (GUI) para Taverna y se utiliza para realizar búsquedas de descripciones de servicios Web a través del *Feta Engine*. Feta permite a los usuarios (Ver ejemplo):

- Crear búsquedas semánticas en un dominio específico de servicios a través del *Feta Engine*.
- Desplegar información acerca de los servicios encontrados
- Integrar los servicios encontrados al diagrama de *workflow* mediante la instrucción **"Add to model"** vista en las secciones anteriores.

Feta se puede iniciar desde Taverna al pulsar la pestaña **"Discover"** (que aparece cuando el software de Feta se ha instalado). Para instalar Feta, se selecciona **"Plugin Manager"** desde el menú **"Tools"** y después se elige **"Find New Plugins"**. Esto se muestra paso a paso en las *Figuras 2.24, 2.25 y 2.26.*

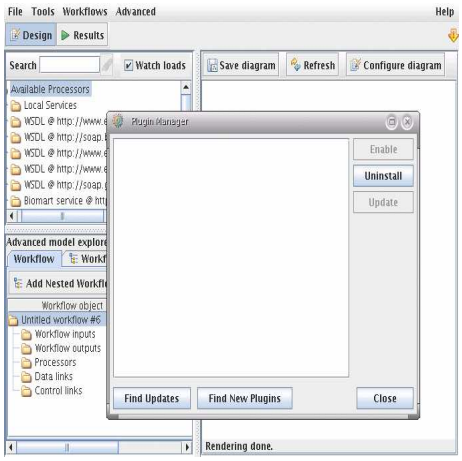


Figura 2.24: Ventana del **Plugin Manager**

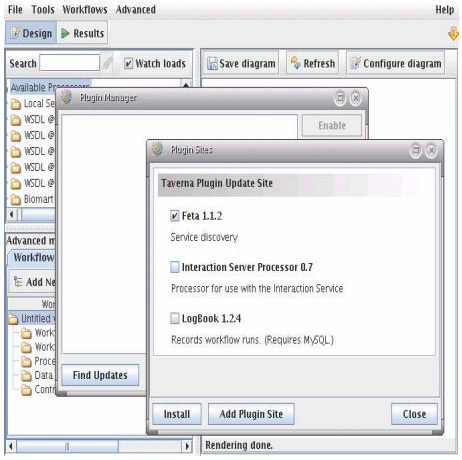


Figura 2.25: Se selecciona el Plugin Feta y se oprime **Install**

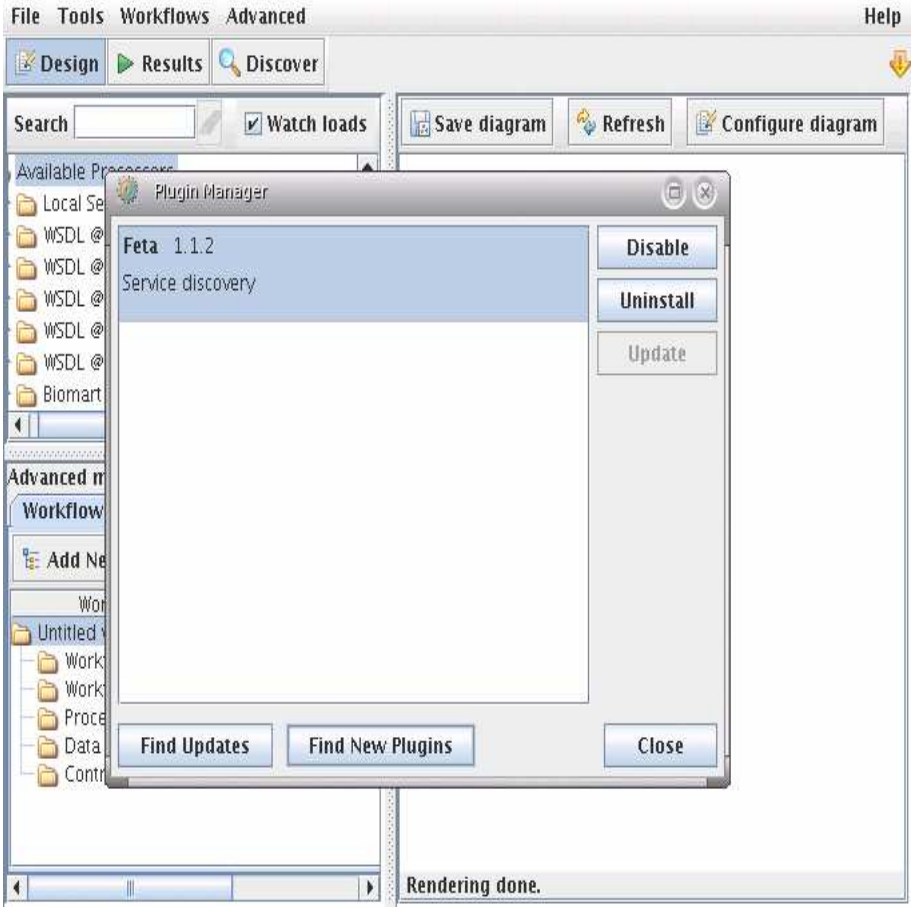


Figura 2.26: Feta se ha instalado

En la *Figura 2.27*, se muestra la interfaz de Feta con la ventana correspondiente a la pestaña *Search services* con los campos que permiten al usuario construir su búsqueda[7]. En la *Figura 2.28*, se aprecia la ventana para la pestaña *Results*. Cualquier información adicional disponible acerca del servicio también es desplegada en la ventana de resultados, permitiendo al usuario hacer la selección final del servicio más apropiado. Estos servicios pueden ser, entonces, añadidos al diagrama de *workflow* mediante la instrucción **"Add to model"**.

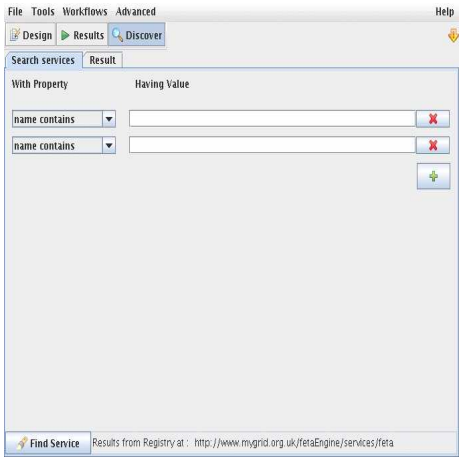


Figura 2.27: Ventana *Search services* de Feta

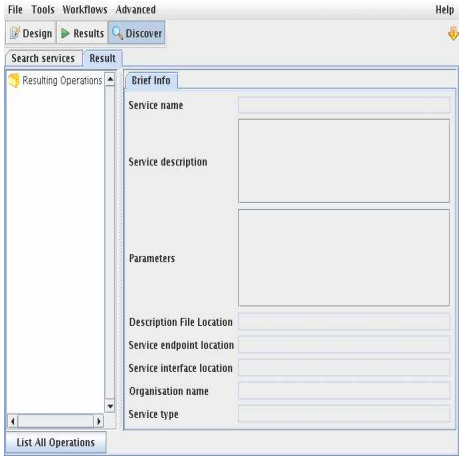


Figura 2.28: Ventana *Result* de Feta

Ejemplo: búsqueda de servicios Web conteniendo el término *Split*

Supongamos que se desea hacer una búsqueda de todos los servicios Web que contengan el término *"Split"*. Para esto, se llenan los campos del panel **Search services** como se muestra en la *Figura 2.29*. El botón con el signo **"+"** de color verde añade un nuevo campo de búsqueda cada vez que se oprime. Los botones con las **"x"** al lado de cada campo se utilizan para borrar. A continuación, se presiona el botón *Find Service* y después de un instante aparece el panel de resultados **Result**. En la *Figura 2.30*, se presentan dos servicios con el término *"Split"*: seqretsplit y splitter. Si se oprime ahora el botón *List All Operations* se mostrarán todos los servicios disponibles. Ver *Figura 2.31*. Para añadir cualquiera de estos servicios a la ventana de diagramas, simplemente se seleccionan y al oprimir el botón derecho del *mouse* se elige **"Add to model"**. En la *Figura 2.32*, se muestra la integración de un par de estos servicios a la ventana de diagramas.

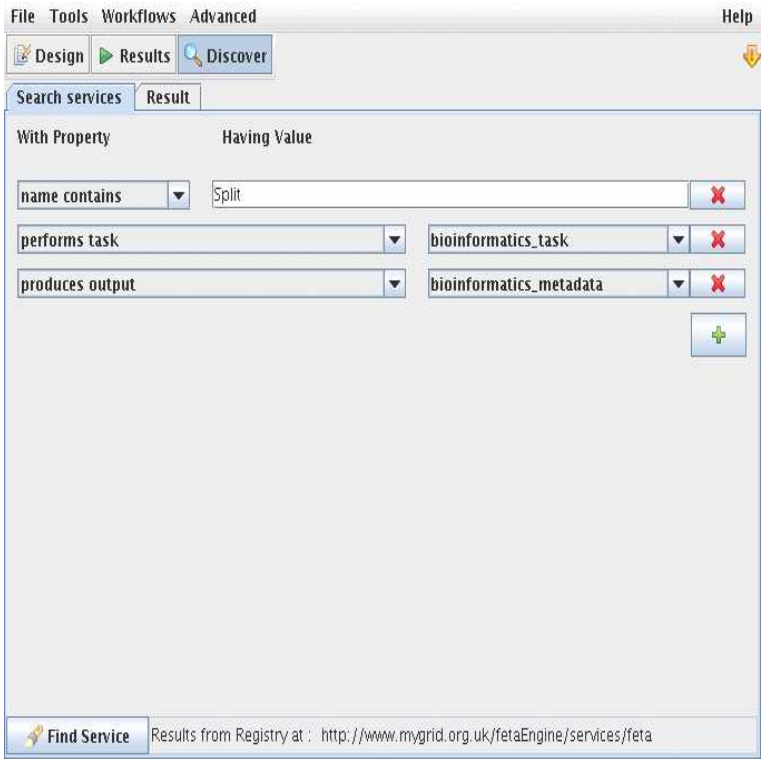


Figura 2.29: Se llenan los campos del panel *Search Services*

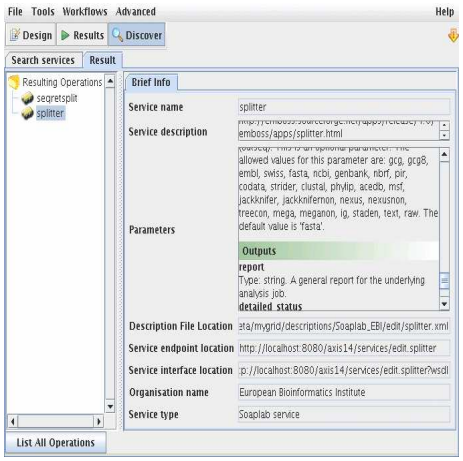


Figura 2.30: Se presentan los resultados de la búsqueda en el panel **Result**

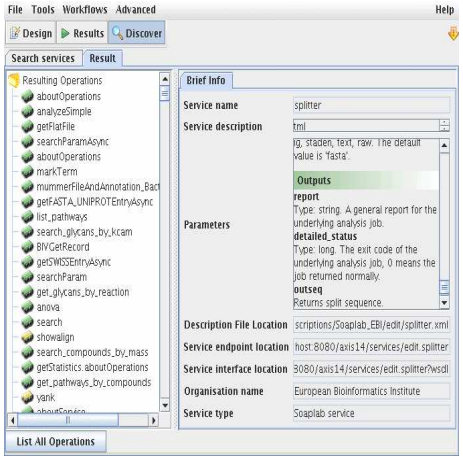


Figura 2.31: Todos los servicios disponibles

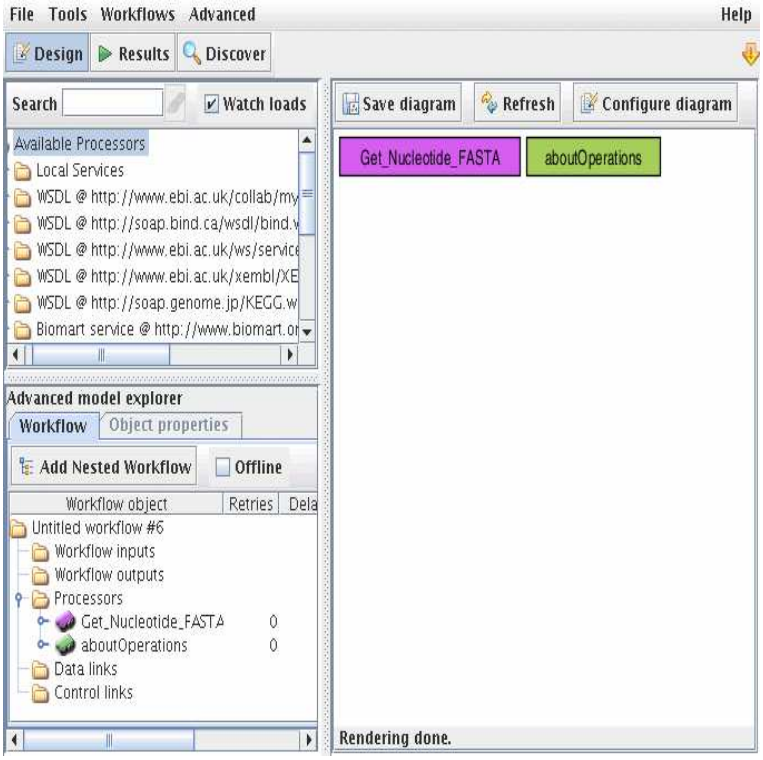


Figura 2.32: Integración de servicios desde Feta a la ventana de diagramas

2.4 Captura y publicación de servicios con PeDRo

PeDRo es una herramienta abierta efectiva para la captura de datos mediante XML Schema[6]. XML Schema es un modelador de documentos basado en el lenguaje de marcado extensible XML (*eXtensible Markup Language*) que permite describir y restringir el contenido de un documento XML.

PeDRo no es una utilidad (plugin) propia de Taverna como Feta, sino un software independiente disponible en el sitio http://pedro.sourceforge.net.

PeDRo es la herramienta preferida entre los usuarios no informáticos para capturar y publicar sus servicios Web debido a su fácil utilización [6]. La interfaz gráfica de usuario de PeDRo presenta plantillas de captura de datos para describir al documento XML. De esta manera, los elementos y atributos capturados se incluyen dentro de un vocabulario organizado (ontología) permitiendo la localización del servicio Web.

Integración de Feta y PeDRo en Taverna

En la *Figura 2.33*, se presenta el diagrama que integra a Feta y PeDRo en el proceso de búsqueda de servicios Web para Taverna[7][9]. El diagrama inicia con el científico que captura y publica sus documentos XML con la asistencia de PeDRo.

Las descripciones de los documentos XML quedan así disponibles en un repositorio de descripción, descubrimiento e integración UDDI (*Universal Description, Discovery and Integration*) de servicios Web. Del otro lado del diagrama, un usuario utiliza la interfaz Feta desde la plataforma de Taverna para construir una petición de búsqueda de servicios Web. Una vez que se inicia la búsqueda, el *Feta Engine* se encarga de encontrar las descripciones en el repositorio que más se acerquen a las solicitadas. Los resultados son, entonces, presentados al usuario.

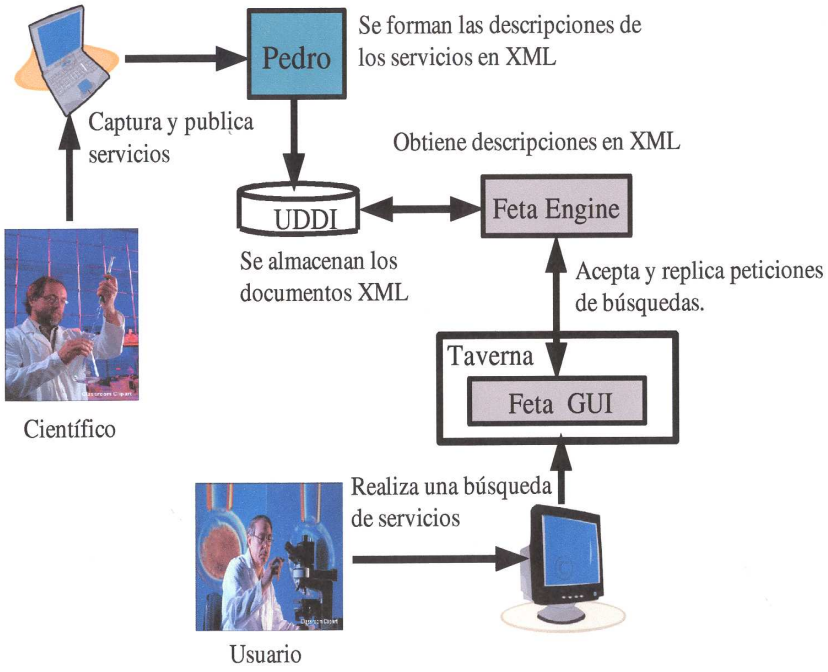


Figura 2.33: Integración de Feta y Pedro

2.5 Sumario

En este capítulo hemos introducido las principales tecnologías y paradigmas relacionados con nuestro trabajo de investigación: servicios Web, *workflows* científicos y ambientes distribuidos gráficos de resolución de problemas. Entre los ambientes distribuidos gráficos de resolución de problemas expuestos en este capítulo, hemos destacado a Taverna como una plataforma bien establecida y ampliamente utilizada para llevar a cabo experimentos con *workflows* en *e-Science*. De esta manera, Tavlab: entorno estadístico basado en *workflows* con Taverna, desarrollado en este trabajo de tesis, aspira a ser una contribución original en lo que a análisis estadístico de datos experimentales se refiere dentro de la *e-Science*. Así, una parte importante

de nuestra investigación se centró en conocer los detalles más importantes de funcionamiento de la plataforma Taverna relevantes a nuestra implementación. Con esto en mente, este capítulo está enfocado primordialmente en describir cómo utilizar la plataforma Taverna.

Capítulo 3

De Taverna a Tavlab

En este capítulo introducimos Tavlab, nuestro entorno para análisis estadístico de datos basado en *workflows*, diseñado e implementado para trabajar dentro del ambiente gráfico de Taverna[1]. En la Sección 3.1, se describe la organización de los recursos de servicios Web con que cuenta nuestro entorno para la resolución de problemas de análisis estadístico de datos en los campos de la Biotecnología, la Química (concretamente la Quimiometría) y la Medicina Estadística. Los recursos de servicios Web están organizados en cuatro niveles dependiendo de las tareas realizadas: administración de los datos de entrada, aplicaciones, apoyo/consulta y presentación de resultados. En la Sección 3.2 se describe el sitio Web, construido dentro del Departamento de Computación, para almacenar y tener disponibles todos los recursos de servicios Web de Tavlab. En seguida, en la Sección 3.3, se explican los aspectos importantes de nuestra implementación como son: la estructura general que siguen los programas de las aplicaciones en lenguaje Java a partir de los cuales se crean los métodos o procedimientos ofrecidos por los servicios Web, así como los estándares y lenguajes propios de los servicios Web. Luego, en la Sección 3.4, por medio de un ejemplo, se introducen los íconos representativos de los diferentes métodos que forman parte de nuestros servicios Web y se explica, mediante un diagrama, la manera de conectar sus entradas y salidas; formando así un *workflow*.

3.1 Recursos de servicios Web en Tavlab

Los recursos de servicios Web de nuestro entorno Tavlab están agrupados en 4 niveles principales de acuerdo a la aplicación para la cual fueron diseñados. A su vez, cada servicio Web cuenta con varios métodos de objetivo específico acorde con la aplicación en curso de ejecución. En la *Figura 3.1* se muestran los distintos servicios Web pertenecientes a cada uno de estos niveles, los cuales se describen a continuación:

[1]Proyecto Taverna, http://taverna. sourceforge.net

3.1.1 Nivel I: manejo de datos de entrada

Este nivel agrupa los servicios diseñados para recibir archivos de datos experimentales. Estos servicios cuentan con métodos para lectura de datos de entrada sin importar su disposición y codificación (ascii o binaria) en el archivo, y los presentan a la entrada de un *workflow* en forma adecuada (arreglo tipo string de datos) para su procesamiento.

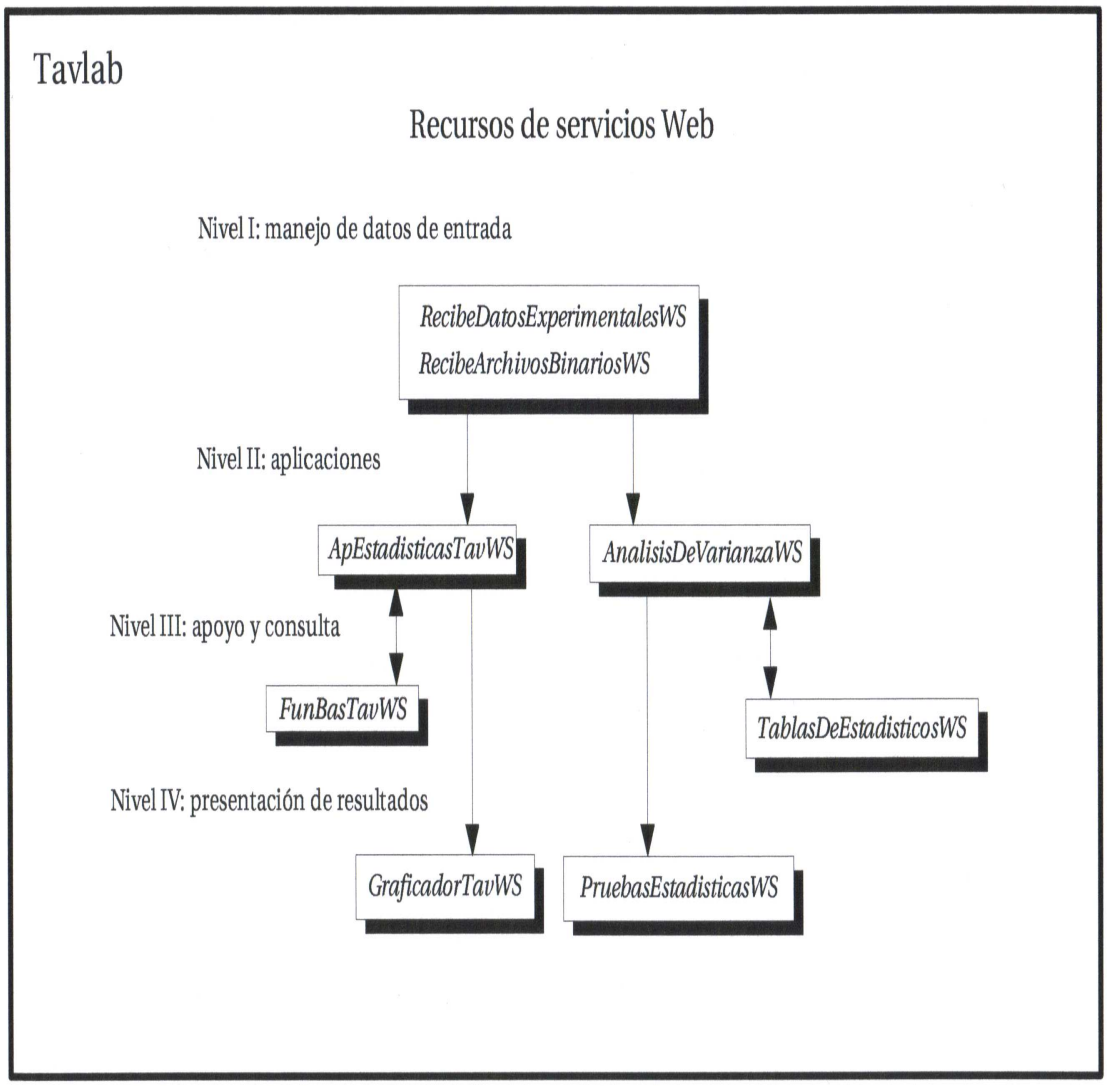


Figura 3.1: Recursos de servicios Web en Tavlab

Los recursos de servicios Web y sus respectivos métodos agrupados en este nivel son los siguientes:

RecibeDatosExperimentalesWS cuenta con 2 métodos que permiten leer datos tipo ascii.

Sus métodos son los siguientes:

1. *recibe_datos:* Lee un archivo de datos sin importar si han sido guardados de manera vertical, horizontal o como matriz. Su salida es un arreglo tipo string de datos.

2. *datos_tratamientos:* Lee un archivo de datos agrupados horizontalmente de acuerdo al número de tratamientos del experimento. Su salida es un arreglo tipo string de datos.

RecibeArchivosBinariosWS está formado por 10 métodos que se encargan de recibir en su entrada archivos binarios de datos y de enviar a su salida el arreglo tipo string de datos correspondiente. De esta manera, el usuario simplemente eligirá el método necesario acorde con las características del archivo binario de datos a analizar.

A continuación, se describe cada método perteneciente a este Servicio Web:

1. *binarioInteger_BigEndian:* Recibe un archivo binario de datos en formato Integer (4-bytes) y sistema Big-Endian. Su salida es el arreglo correspondiente tipo string de datos.

2. *binarioInteger_LittleEndian:* Recibe un archivo binario de datos en formato Integer (4-bytes) y sistema Little-Endian. Su salida es el arreglo correspondiente tipo string de datos.

3. *binarioShort_BigEndian:* Recibe un archivo binario de datos en formato Short (2-bytes) y sistema Big-Endian. Su salida es el arreglo correspondiente tipo string de datos.

4. *binarioShort_LittleEndian:* Recibe un archivo binario de datos en formato Short (2-bytes) y sistema Little-Endian. Su salida es el arreglo correspondiente tipo string de datos.

5. *binarioLong_BigEndian:* Recibe un archivo binario de datos en formato Long (8-bytes) y sistema Big-Endian. Su salida es el arreglo correspondiente tipo string de datos.

6. *binarioLong_LittleEndian:* Recibe un archivo binario de datos en formato Long (8-bytes) y sistema Little-Endian. Su salida es el arreglo correspondiente tipo string de datos.

7. *binarioFloat_BigEndian_IEEE754:* Recibe un archivo binario de datos en formato Float (4-bytes) ya sea en sistema Big-Endian o IEEE754. Su salida es el arreglo correspondiente tipo string de datos.

8. *binarioFloat_LittleEndian:* Recibe un archivo binario de datos en formato Float (4-bytes) y sistema Little-Endian. Su salida es el arreglo correspondiente tipo string de datos.

9. *binarioDouble_BigEndian_IEEE754:* Recibe un archivo binario de datos en formato Double (8-bytes) ya sea en sistema Big-Endian o IEEE754. Su salida es el arreglo correspondiente tipo string de datos.

10. *binarioDouble_LittleEndian:* Recibe un archivo binario de datos en formato Double (8-bytes) y sistema Big-Endian. Su salida es el arreglo correspondiente tipo string de datos.

Como un ejemplo representativo de los servicios del nivel I, la *Figura 3.2* ilustra la aplicación del método *recibe_datos* perteneciente al servicio *RecibeDatosExperimentalesWS*. En la *Figura 3.2*, se utiliza un archivo de datos plano llamado *datos.txt* para suministrar los datos de entrada al *workflow*. Sin embargo, debido a que este archivo contiene, además de los datos, espacios ($\backslash\backslash s$) y saltos de línea ($\backslash\backslash r$), la entrada del método *recibe_datos* incluirá los strings $\backslash\backslash s$ y $\backslash\backslash r$ como se muestra en el arreglo tipo strings de entrada. El método *recibe_datos* se encargará, entonces, de eliminar los espacios ($\backslash\backslash s$) y saltos de línea ($\backslash\backslash r$), y construirá un nuevo arreglo tipo strings conteniendo sólo los datos. Esta es la forma apropiada de enviar grupos de datos a los métodos de los servicios pertenecientes a los siguientes niveles de nuestro entorno Tavlab. Así, por ejemplo, se muestra a *recibe_datos* suministrando el arreglo tipo strings de datos, sin espacios y saltos de línea, al método *media* perteneciente al nivel II.

De esta manera, los métodos del nivel I leen datos de entrada sin importar su disposición y codificación (ascii o binaria) en el archivo, y los presentan a la entrada de un *workflow* en forma adecuada (arreglo tipo string de datos) para su procesamiento.

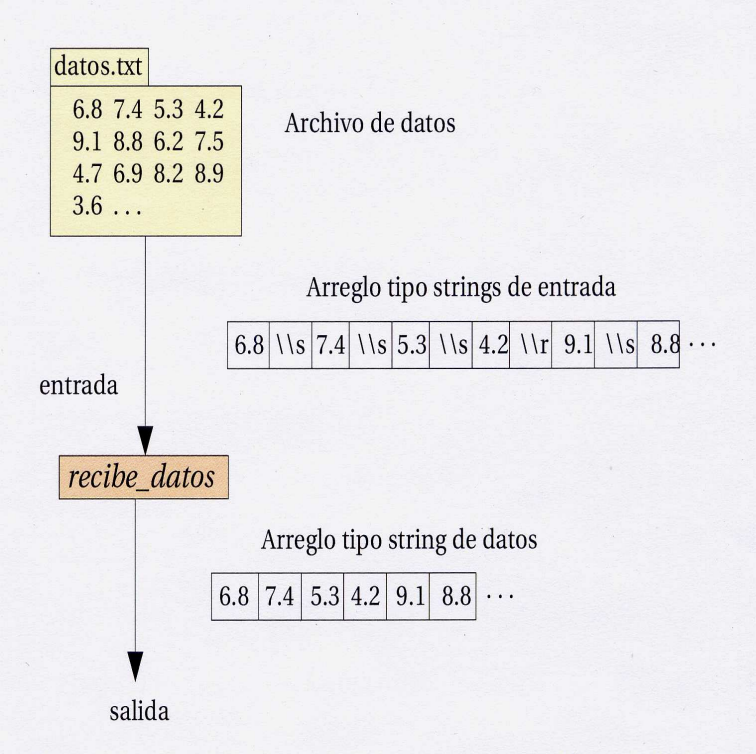


Figura 3.2: Aplicación del método *recibe_datos* perteneciente al nivel I

3.1.2 Nivel II: aplicaciones

En este nivel se encuentran los servicios que se diseñaron para realizar los cálculos del análisis estadístico de datos en los campos de la Químiometría y la Medicina Estadística, así como del análisis de varianza en el área de la Biotecnología.

Los recursos de servicios Web y sus respectivos métodos agrupados en este nivel son los siguientes:

ApEstadisticasWS contiene 12 métodos para calcular las medidas de tendencia central y de dispersión, la ecuación de regresión, el coeficiente de correlación y la prueba t de Student. El arreglo tipo string de datos con que se alimentan estos métodos corresponden a los datos de entrada previamente procesados por los métodos del nivel I. Luego, cada método alimentado con este arreglo convierte los datos tipo string en tipo double para realizar sus respectivos cálculos.

A continuación, se describe cada método perteneciente a este servicio Web:

1. *media:* recibe un arreglo tipo string de datos y calcula la media.
2. *mediana:* recibe un arreglo tipo string de datos y obtiene la mediana.
3. *moda:* recibe un arreglo tipo string de datos y encuentra la moda. Si el conjunto de datos no tiene moda, envía el mensaje "El conjunto de datos no tiene moda".
4. *varianza:* recibe un arreglo tipo string de datos, así como un dato tipo double correspondiente al valor de la media, y calcula la varianza.
5. *var_conjunta:* recibe dos arreglos tipo string de datos, así como los valores de las varianzas correspondientes a cada conjunto de datos, y calcula la varianza conjunta.
6. *ecuReg:* recibe los valores de las medias de dos conjuntos de datos, así como el valor del coeficiente de regresión correspondiente a ambos conjuntos de datos, y obtiene la ecuación de regresión.
7. *coeCorr:* recibe un arreglo tipo string de datos correspondiente a uno de dos conjuntos de datos de entrada, así como los valores de las medias y desviaciones estándar de ambos conjuntos, y encuentra el coeficiente de correlación.
8. *coeReg:* recibe los valores de las desviaciones estándar correspondientes a dos conjuntos de datos de entrada, así como el valor del coeficiente de correlación de ambos conjuntos, y encuentra el coeficiente de regresión.
9. *student_t_Vc:* recibe los valores de las medias de dos conjuntos de datos, así como el valor de la varianza conjunta de ambos conjuntos, y encuentra el valor t de Student.
10. *student_t_Corr:* recibe un arreglo tipo string de datos, así como el valor del coeficiente de correlación, y obtiene el valor t de Student correspondiente a la correlación.
11. *desvResEst:* recibe un arreglo tipo string de datos, así como los valores del cuadrado del coeficiente de correlación y la varianza, y calcula la desviación residual estándar.
12. *errorEst_m:* recibe un arreglo tipo string de datos, así como los valores de la media y la desviación residual estándar, y obtiene el error estimado de la pendiente de la línea de regresión.

AnalisisDeVarianzaWS cuenta con 6 métodos diseñados para realizar el análisis de varianza y aplicar las pruebas estadísticas de comparaciones de pares de medias de Tukey y del LSD (Less Significant Difference) de Fisher en Biotecnología.

A continuación, se describe cada método perteneciente a este servicio Web:

1. *grados_libertad:* recibe un arreglo tipo string de datos y obtiene los grados de libertad.
2. *suma_cuadrados:* recibe un arreglo tipo string de datos y calcula la suma de cuadrados.
3. *medias_tratamientos:* recibe un arreglo tipo string de datos y obtiene las medias de los tratamientos.
4. *cuadrado_medio:* recibe los valores correspondientes a los grados de libertad y la suma de cuadrados y calcula el cuadrado medio.
5. *estadistico_Fo:* recibe el valor del cuadrado medio y obtiene el estadístico de prueba F_0.
6. *tabla_AV:* recibe los valores correspondientes a los grados de libertad, la suma de cuadrados y el estadístico de prueba F_0 y grafica la tabla de análisis de varianza.

Como un ejemplo representativo de los servicios del nivel II, la *Figura 3.3* ilustra la aplicación de los métodos *media* y *varianza* pertenecientes al servicio *ApEstadisticasTavWS*. Junto con el método *recibe_datos* del nivel I, se construye el *workflow* para calcular la varianza de los datos de entrada. La *Figura 3.3* muestra la ecuación para calcular la media (M) implementada por el método *media*, así como la ecuación para calcular la varianza (V) implementada por el método *varianza*. De la ecuación para la varianza se aprecia que el valor de la media es requerido por el método *varianza* previo a realizar los cálculos. Así, el método *media* se encarga de calcular y suministrar el valor de la media a *varianza* en un paso anterior.

De esta manera, en el nivel II se encuentran los servicios Web con los métodos correspondientes a las aplicaciones implementadas propias de la estadística. Así, nuevos servicios Web con métodos para realizar futuras aplicaciones se incluirán en el nivel II.

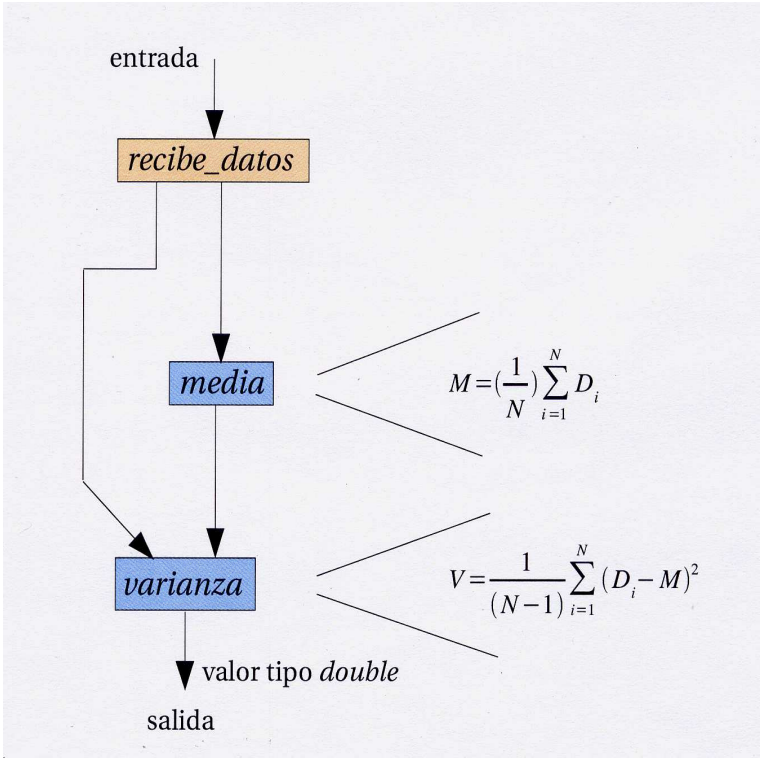


Figura 3.3: Aplicación de los métodos *media* y *varianza* pertenecientes al nivel II

3.1.3 Nivel III: apoyo y consulta

Debido a que las operaciones aritméticas de la raíz cuadrada, elevar al cuadrado y valor absoluto con frecuencia se utilizan en los cálculos estadísticos, por ejemplo, la desviación estándar se obtiene al aplicar la raíz cuadrada a la varianza, se crearon métodos para estas operaciones en este nivel. Asimismo, esta organización permite que la construcción de *workflows* estadísticos sea más clara y directa. Por otro lado, en este mismo nivel se incluyeron los servicios que implementan las tablas estadísticas empledas en los cálculos de análisis de varianza y en las pruebas de Tukey y de LSD de Fisher.

Los recursos de servicios Web y sus respectivos métodos agrupados en este nivel son los siguientes:

FunBasTavWS cuenta con 3 métodos diseñados para obtener la raíz cuadrada (raizCua), la segunda potencia (elevarCua) y el valor absoluto (valorAbs) de un dato tipo double.

TablasDeEstadisticosWS contiene los métodos que implementan las tablas estadísticas correspondientes a los puntos porcentuales de la distribución **t** y a los puntos porcentuales del estadístico de rango studentizado **q**, utilizadas en el análisis de varianza y en las pruebas de comparaciones de pares de medias de Tukey y Fisher.

A continuación, se describe cada método perteneciente a este Servicio Web:

1. *tabla_q05:* regresa el valor del estadístico **q** para $\alpha = 0.05$ a partir de los datos experimentales de entrada.
2. *tabla_t:* regresa el valor del estadístico **t** a partir del parámetro α y los grados de libertad.

Como un ejemplo representativo de los servicios del nivel III, la *Figura 3.4* ilustra la aplicación del método *raizCua* perteneciente al servicio *FunBasTavWS*. El método *raizCua* es añadido a la salida del *workflow* de la *Figura 3.3* con la finalidad de obtener la desviación estándar de los datos de entrada. La *Figura 3.4* muestra la ecuación para obtener la desviación estándar (DE) a partir de extraer la raíz cuadrada del valor de la varianza previamente calculado por *varianza*.

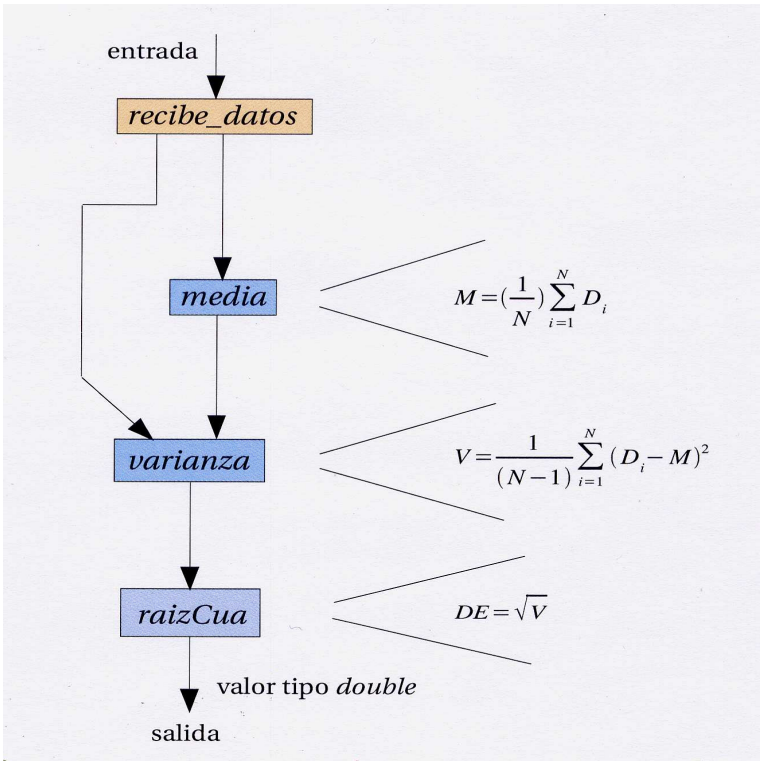


Figura 3.4: Aplicación del método *raizCua* perteneciente al nivel III

3.1.4 Nivel IV: presentación de resultados

En este nivel se encuentran los servicios destinados a graficar la línea de regresión, así como las tablas de resultados correspondientes a las pruebas de Tukey y del LSD de Fisher.

Los recursos de servicios Web y sus respectivos métodos agrupados en este nivel son los siguientes:

GraficadorTavWS cuenta con el método *graficar_ER* para graficar la línea de regresión a partir de dos conjuntos de datos de entrada y la ecuación de regresión.

PruebasEstadisticasWS Está formado por los métodos que grafican las tablas de resultados correspondientes a las pruebas de Tukey y Fisher.

A continuación, se describe cada método perteneciente a este Servicio Web:

1. *prueba_Tukey:* grafica la tabla de resultados de la prueba de Tukey a partir de los grados de libertad, la suma de cuadrados, las medias de los tratamientos y del estadístico **q**.
2. *prueba_LSD:* grafica la tabla de resultados de la prueba del LSD de Fisher a partir de los grados de libertad, la suma de cuadrados, las medias de los tratamientos y del estadístico **t**.

Como un ejemplo representativo de los servicios del nivel IV, la *Figura 3.5* ilustra la aplicación del método *graficar_ER* perteneciente al servicio *GraficadorTavWS*. La *Figura 3.5* muestra sólo la parte final del *workflow* que grafica la línea de regresión. Para graficar la línea de regresión sobre la dispersión de los datos de entrada, se requiere suministrar los datos de entrada X, los datos de entrada Y y la expresión correspondiente a la ecuación de regresión a la entrada del método *graficar_ER*. Los datos de entrada X e Y le son suministrados por medio del método *recibe_datos* del nivel I, y la expresión para la ecuación de regresión por medio del método *ecuReg* del nivel II. El método *ecuReg*, a su vez, requerirá de datos en sus entradas provenientes de otros métodos en la parte superior del *workflow* como son: la media de los datos X (Mx), la media de los datos Y (My) y el coeficiente de correlación (r). El método *graficar_ER*, entregará a su salida un arreglo de datos tipo *byte* correspondiente a la gráfica.

De esta manera, el nivel IV incluirá todos los servicios Web con métodos para presentar resultados gráficamente.

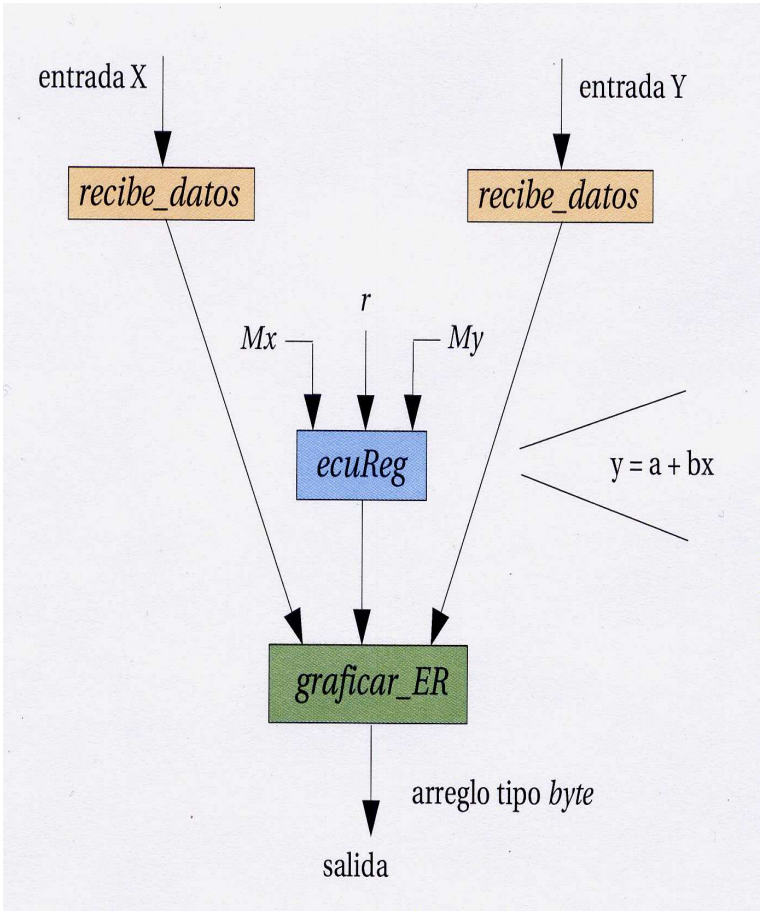


Figura 3.5: Aplicación del método *graficar_ER* perteneciente al nivel IV

3.2 Servidor de recursos en Tavlab

Con el fin de almacenar y tener disponibles todas nuestras aplicaciones, se instaló un sitio Web para Tavlab en el Departamento de Computación. La dirección de este sitio Web es http://www.tavlab-cinvestav.mx:2112/. La *Figura 3.6* muestra cómo una terminal de usuario con Taverna (una laptop, por ejemplo), puede acceder a los recursos de servicios Web de nuestro entorno Tavlab (http://www.tavlab-cinvestav.mx:2112) e incluso acceder archivos de datos desde un sitio Web diferente (http://www.matrix-cinvestav.mx/∼urevilla/data) para alimentar nuestras aplicaciones.

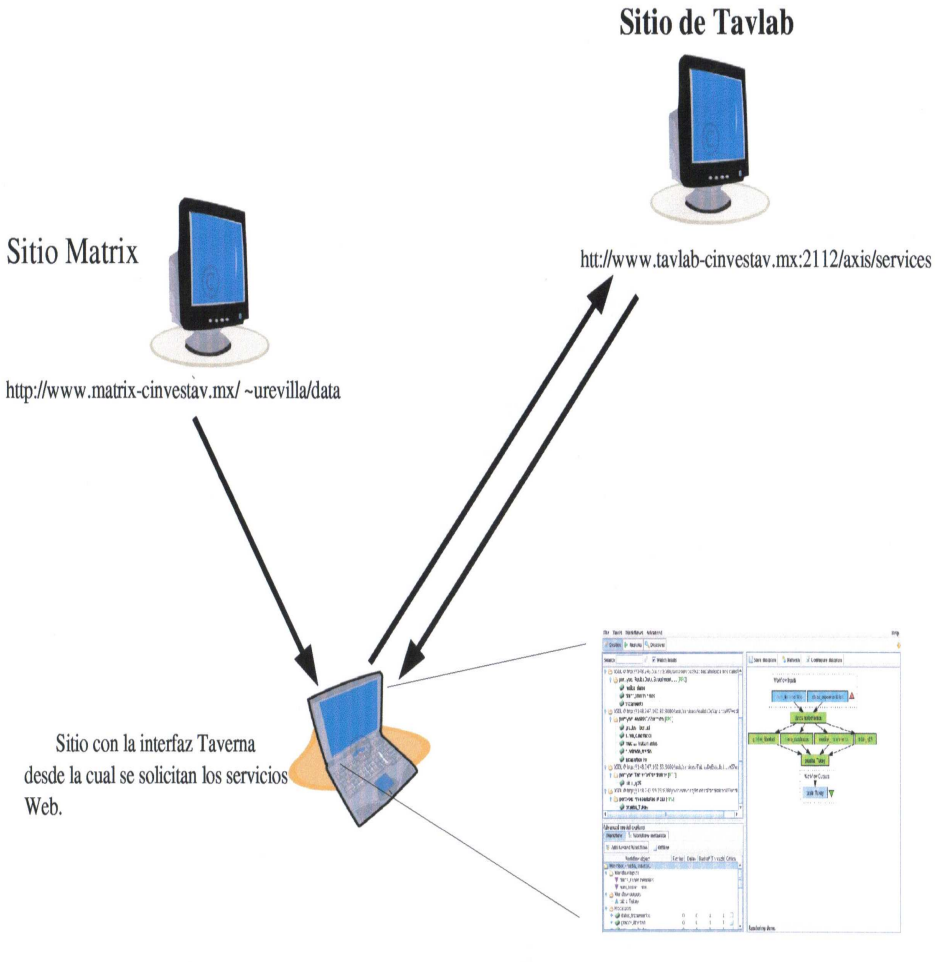


Figura 3.6: Sitio Web para nuestras aplicaciones de Tavlab

3.3 Implementación de los recursos de servicios Web

En esta sección se describe, en primer lugar, la estructura general de los programas escritos en lenguaje Java a partir de los cuales se obtienen las clases correspondientes a nuestros servicios Web antes descritos. Luego, con el fin de explicar los documentos descriptores *WSDD* y *WSDL* utilizados en la creación de nuestros servicios Web, se expondrán brevemente los protocolos y lenguajes empleados para construir estos documentos. En seguida, se explicará el proceso de obtención de las clases Java que hacen de proxy. Un proxy es un programa que provee recursos localizados en máquinas remotas conectándose al servidor especificado. Cuando un equipo conectado a Internet desea acceder a una información o recurso, el proxy es quien realiza la comunicación y a continuación traslada el resultado al equipo inicial. Para finalizar esta sección, se explicará la forma de interconectar las entradas y salidas de íconos representativos de los métodos pertenecientes a los diferentes servicios Web implementados.

3.3.1 Estructura general de los programas

A continuación, se describe la estructura general de los programas desarrollados en lenguaje Java para los recursos de servicios Web. Esta estructura general está dividida en 4 secciones:

La primera sección de esta estructura, corresponde a las librerías y/o paquetes de Java que cada clase necesita dependiendo de la aplicación a la que esté dedicada.

La segunda sección, está destinada a la definición de las excepciones requeridas. Puede ser que durante el diseño de un método para determinada aplicación se decida atrapar (*catch*) un determinado error (excepción) y enviar el mensaje correspondiente. Por ejemplo, para el método *moda*, se definió una excepción para enviar el mensaje: *"El conjunto de datos no contiene moda"*, cuando los datos de entrada no cuentan con un valor que se repita.

La tercera sección, se refiere al nombre de la clase, el cual debe ser igual al nombre del servicio Web a crear.

La cuarta y última sección, corresponde a la definición de los métodos ofrecidos por el servicio Web.

Es importante señalar que los arreglos de datos enviados de un procedimiento (método) a otro deben ser siempre de tipo string, y sólo los datos enviados individualmente pueden ser de cualquier otro tipo (integer, double, float, etc).

Así también, cada método debe contar con una instrucción ***return*** para enviar el resultado de los datos procesados a la aplicación solicitante.

import java.io.;* */*PRIMERA SECCIÓN*/*
import java.util.;*
import java.awt.;*
import java.text.;*

. . . otras librerías o paquetes (packages) Java.

/ SEGUNDA SECCIÓN */*

*** Espacio para la definición de excepciones ***

/ TERCERA SECCIÓN */*

public class ***Nombre del Servicio Web*** {

/ CUARTA SECCIÓN */*

public ***tipo de dato*** **método1***(String[] args) throws IOException, . . . posiblemente otras excepciones*{
tipo de dato *var1;*
*** Instrucciones para el* **método1** ****
return *var1*
}*** Termina el* **método1** ****

public ***tipo de dato*** **método2***(String[] args) throws IOException, . . . posiblemente otras excepciones*{
tipo de dato *var2;*
*** Instrucciones para el* **método2** ****
return *var2*
}*** Termina el* **método2** ****

public ***tipo de dato*** **métodoN***(String[] args) throws IOException, . . . posiblemente otras excepciones*{
tipo de dato *varN;*
*** Instrucciones para el* **métodoN** ****
return *varN*
}*** Termina el* **métodoN** ****
}*** Termina la* **clase principal** ****

Además, los métodos no podrán incluir instrucciones para imprimir en pantalla, como son las que inician con ***System.out***. Esto debido, por una parte, a que tales instrucciones producen una salida sólo en el sitio donde se ejecutan los procedimientos, y por otra, a que el documento WSDL que describe al servicio Web producirá errores, ya que tales instrucciones no tienen significado para la aplicación cliente. De esta manera, para enviar un resultado o mensaje a la aplicación cliente, siempre se deberá utilizar la instrucción ***return***.

3.3.2 Estándares y lenguajes de los servicios Web

Para crear los servicios Web es imprescindible conocer las diferentes tecnologías involucradas. En seguida presentamos una breve descripción de los protocolos y lenguajes que hacen posible la funcionalidad de nuestros servicios Web.

Lenguaje de marcado extensible: XML (*eXtensive Markup Language*)

XML surge como un estándar para el intercambio de información estructurada entre diferentes plataformas, permitiendo tener acceso a datos en cualquier parte de la Internet.

XML es un metalenguaje extensible de etiquetas desarrollado por el *World Wide Web consortium* (W3C). Es una simplificación y adaptación del SGML (Structured Generalized Markup Language) un estándar ISO para documentos estructurados sumamente complejo para poder servir documentos en la Web. XML permite definir la gramática de lenguajes específicos, siendo un formato estándar para definir lenguajes [21].

La comunicación entre sistemas de información existentes, e incluso entre partes de un mismo sistema, suele convertirse en un problema, puesto que generalmente coexisten aplicaciones desarrolladas en lenguajes, bases de datos, plataformas e incluso, protocolos de comunicación diferentes. Allí surge la necesidad de unificar el formato de envío y recepción de información: XML proporciona reglas claras para estructurar documentos. De esta manera cuando los documentos viajan através de una red, transportamos sólo las estructuras y los datos contenidos. No es necesario intercambiar herramientas para interpretar dicha información: muchos lenguajes traen incorporadas las suyas propias. Al existir la posibilidad de estructurar la información según nuestra necesidad, las búsquedas de los documentos serán más específicas y coherentes.

Llamadas a procedimientos remotos: RPC (*Remote Procedure Calls*)

RPC es un protocolo utilizado para establecer conexiones y facilitar transacciones entre dos sistemas remotos. Una llamada a procedimiento remoto (RPC) se inicia cuando el cliente envía un mensaje de petición a un servidor remoto conocido para ejecutar un programa específicado por los parámetros suministrados en el mismo mensaje. Sin embargo, las aplicaciones que utilizan RPC para comunicarse no pueden hacerlo mediante HTTP (RPC no fue diseñado para eso).

En contraste, XML-RPC es una implementación basada en el modelo RPC que permite transportar datos codificados en XML entre dos máquinas remotas utilizando HTTP (considerado el primer protocolo de comunicación con estas características [21]). Los servicios Web son considerados como el siguiente paso en la evolución a partir de RPC y XML-RPC.

Protocolo simple de acceso a objetos: SOAP (*Simple Object Access Protocol*)

SOAP es el protocolo de intercambio de mensajes utilizado por los servicios Web para comunicarse. Se deriva del protocolo XML-RPC, y es independiente de la plataforma y del lenguaje de programación en el que se desarrollen las aplicaciones que intentan comunicarse. Cada mensaje se usa como un *envelope* (sobre), el cual contiene dos partes: el *header* (cabecera) y el *body* (cuerpo). El *header* contiene información de control acerca de distintos aspectos del mensaje, por ejemplo, destinatario, posible receptor y un conjunto de opciones de entrega (como la información acerca de la encriptación utilizada). El *body* guarda el contenido del mensaje, normalmente consistente de una petición/pregunta o una contestación/respuesta. El *header* es opcional mientras que el *body* es obligatorio [22]. Tanto el *header* como el *body* pueden tener múltiples subpartes en forma de bloques llamados hijos (*Figura 3.7*). SOAP asume que cada mensaje cuenta con un remitente, un destinatario y un número arbitrario de intermediarios (nodos) para procesar el mensaje y enrutarlo hacia el receptor.

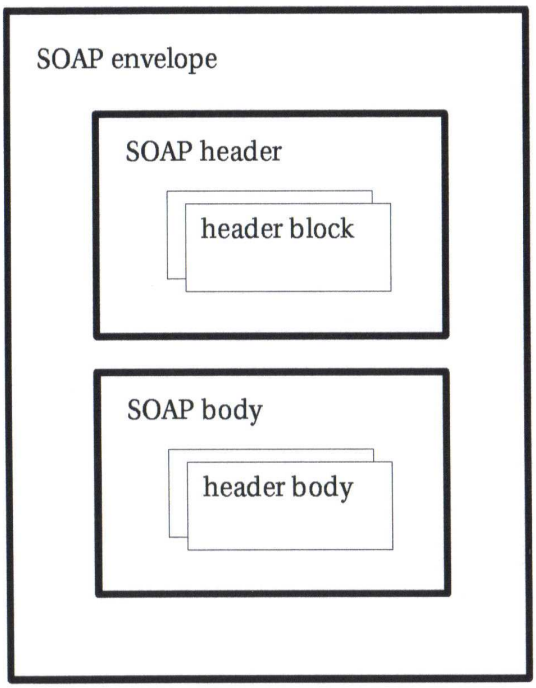


Figura 3.7: Representación de un mensaje SOAP

El protocolo SOAP se usa en dos estilos de interacción: *document-style* (estilo de documento) y RPC-*style*. En el *document-style*, se envía el documento con la orden de la acción a realizar y se recibe un *acknowledgement* (verificación). En el RPC-*style* se incluye el nombre del procedimiento (método) a invocar y los parámetros de entrada, recibiéndose el resultado y los parámetros de salida [22]. De esta manera, el estilo de interacción para los mensajes SOAP utilizados es el RPC-*style*, el cual se muestra en la *Figura 3.8*. Así también, SOAP (a partir de la versión 1.2) define una forma de codificación llamada *SOAP encoding*. Las estructuras de datos, incluyendo tipos básicos tales como *integers* y *strings*, así como tipos complejos como *arrays* y *structures*, pueden ser codificados en forma de instrucciones XML.

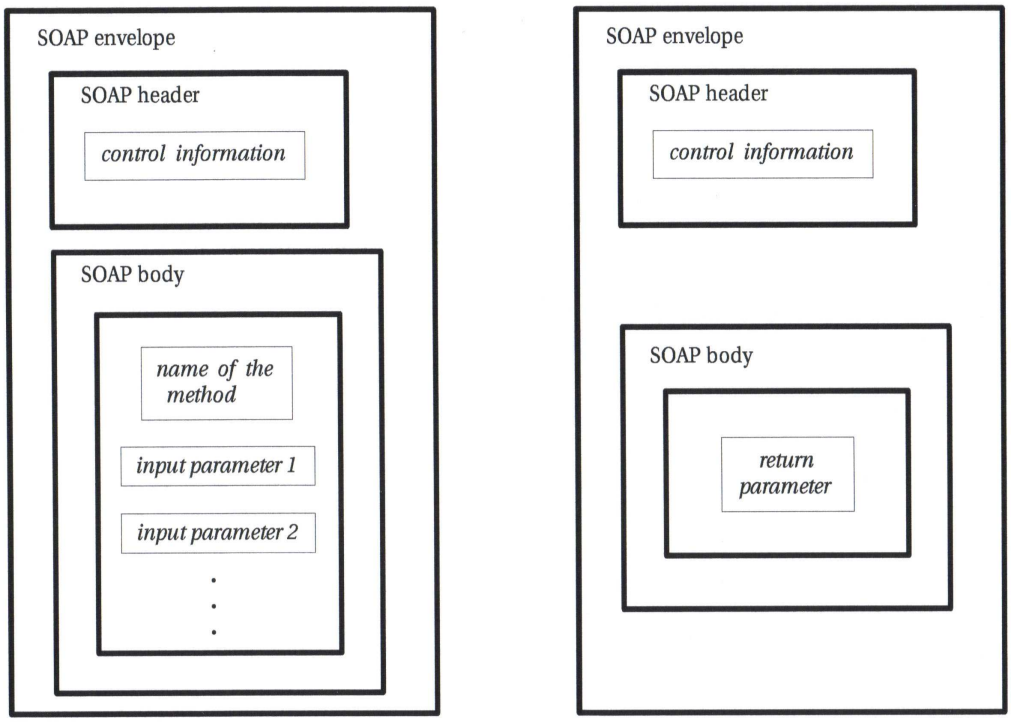


Figura 3.8: Estructura de un mensaje SOAP/RCP-style

SOAP no está ligado a un protocolo de transporte específico: sólo necesita de la capacidad de transportar texto plano (los mensajes son documentos XML, y éstos están formados por texto). Generalmente, a SOAP y HTTP se les asocia, pero esto se deriva únicamente de la masiva utilización de ambas tecnologías (por separado o en conjunto). Cuando se utiliza HTTP como el protocolo de transporte, la URL del recurso que se desea invocar describe al receptor del mensaje SOAP. La *Figura 3.9* muestra mensajes SOAP envueltos por el protocolo HTTP y viajando del transmisor al receptor y viceversa a través de Internet. Esta es la forma en que se lleva a cabo

la comunicación con nuestros servicios Web.

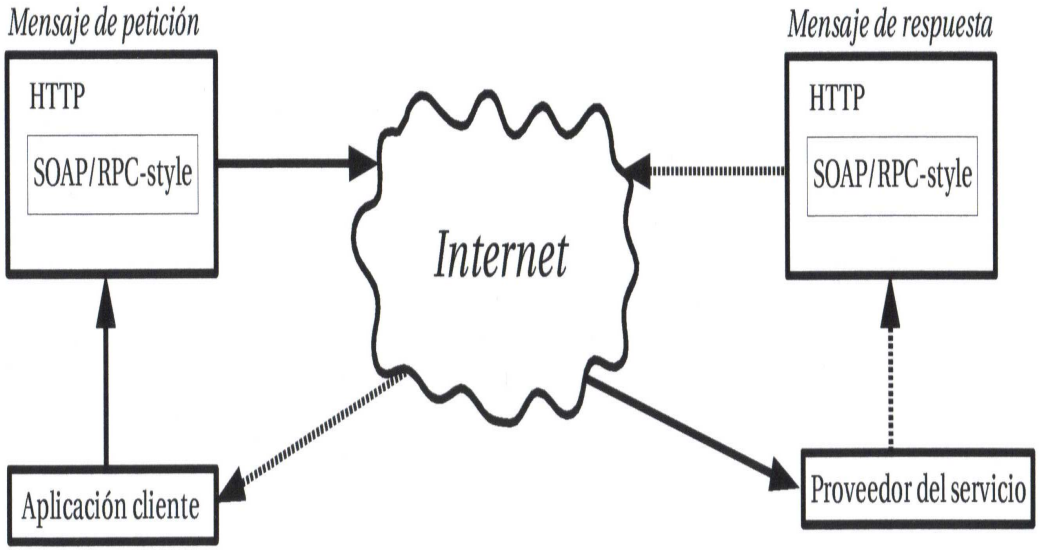


Figura 3.9: Comunicación de mensajes SOAP/RPC-style mediante HTTP

Descriptor de activación de servicios Web: WSDD (*Web Services Deployment Descriptor*)

WSDD es un documento basado en XML utilizado para describir la manera en que los componentes de Axis deben de activarse para procesar los mensajes que entran y salen del servicio. Axis es una plataforma abierta para la implementación de servicios Web basada en XML perteneciente a la *Apache Software Foundation* previamente instalada en el servidor con todas nuestras aplicaciones. El formato de WSDD está definido por el protocolo SOAP v.2.0. La primera línea del documento WSDD contiene el elemento $< deployment >$ para indicar al motor de Axis la activación de un servicio o el elemento $< undeployment >$ para notificar al motor de Axis la desactivación de un servicio en particular. Como ejemplo, se muestra en la *Figura 3.10* el documento WSDD correspondiente al servicio Web *GraficadorTavWS*, y se describe a continuación.

```
<deployment xmlns="http://xml.apache.org/axis/wsdd/"
    xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">
  <service name="GraficadorTavWS" provider="java:RPC">
    <parametre name="className" value="GraficadorTav"/>
    <parameter name="allowedMethods" value="*"/>
  </service>
</deployment>
```

Figura 3.10: Documento WSDD correspondiente al servicio Web *GraficadorTavWS*

La primera línea de este documento indica al motor de Axis que se trata de una activación. Mediante las siguientes líneas definimos el servicio Web a activar *GraficadorTavWS*, el protocolo de comunicación *RPC*, el nombre de la clase que implementa los métodos expuestos *GraficadorTav* y los métodos que serán visibles desde el exterior. El símbolo * indica que todos los métodos del servicio serán visibles.

Lenguaje de descripción de servicios Web: WSDL (*Web Services Description Language*)

WSDL es el documento encargado de describir el servicio Web y está escrito en XML. WSDL indica cómo interactuar con el servicio, los datos necesarios a ser enviados, los datos esperados como respuesta, las operaciones que tienen lugar, el formato y el protocolo necesarios para invocar el servicio, así como su localización [22].

El documento WSDL está formado de tres partes principales: una parte destinada a la declaración de los nombres de espacio (namespace), una parte abstracta y una parte concreta [22]. Los nombres de espacio en XML denotan conjuntos de vocabularios que difieren semánticamente dependiendo del contexto de la aplicación considerada [21]. Un documento XML puede contener nombres de elementos pertenecientes a más de un vocabulario. Si cada vocabulario se vincula con un nombre de espacio, entonces la ambiguedad entre nombres de elementos idénticos queda resuelta. La declaración del nombre de espacio utiliza la siguiente sintaxis:

xmlns=“url”

En donde:

xmlns: notifica al parser XML una declaración de espacio de nombres.
url: corresponde a la dirección URL del sitio en Internet donde se encuentra el espacio de nombres.

La parte abstracta se compone de definiciones *port type* (tipo de puerto), *operations* (operaciones), *messages* (mensajes) y *types* (tipos). Cada "tipo de puerto" es un conjuto de operaciones interrelacionadas ofrecidas por el servicio. Las "operaciones" ofrecidas por el servicio pueden ser [21]:

- *One-way*: la operación recibe mensajes pero no da una respuesta.
- *Request-response*: la operación recibe peticiones y da una respuesta.
- *Solicit-response*: la operación envia una petición y espera por una respuesta.
- *Notification*: la operación envia una petición pero no espera por una respuesta.

Un "mensaje" es una unidad de comunicación con el servicio Web y representa el intercambio de datos en una transmisión simple [22]. Los "tipos" definen a los datos utilizados por el servicio (dato tipo string, integer etc).

La parte concreta del documento WSDL se compone de los elementos *bindings* (enlaces), *port* (puerto) y *services* (servicios). El elemento *"bindings"* se refiere a los protocolos de comunicación utilizados por el servicio.

El "puerto" se refiere al último receptor del mensaje de petición que se define mediante una URL y una dirección de red. Los "servicios" son agrupamientos lógicos de "puertos".

Como ejemplo de la combinación de los componentes de la parte abstracta y la parte concreta, la *Figura 3.11* describe el documento WSDL correspondiente al servicio Web *GraficadorTavWS*.

La primera parte está formada por los nombres de espacio (namespace), los cuales introducen las direcciones URL donde se localizan los diferentes vocabularios requeridos por el documento. El siguiente bloque correspondiente a la parte abstracta contiene los "mensajes", "tipo de puerto" y las "operaciones". Hay dos tipos de mensajes: el mensaje con los datos (número y tipo) que necesitan enviarse al método particular *graficar_ER* llamado *graficar_ERRequest* y el mensaje con los datos enviado como respuesta por el mismo método llamado *graficar_ERResponse*. El "tipo de puerto" con nombre *GraficadorTav* ofrece la "operación" *graficar_ER*. El último bloque encierra los elementos de la parte concreta, los cualos son: el protocolo de comunicación (*"binding"*), el "servicio" y el "puerto". El protocolo de comunicación utilizado es el de llamadas a procedimientos remotos (RPC). El "servicio" consiste de un solo "puerto", el cual especifica la ubicación de la aplicación mediante su dirección URL.

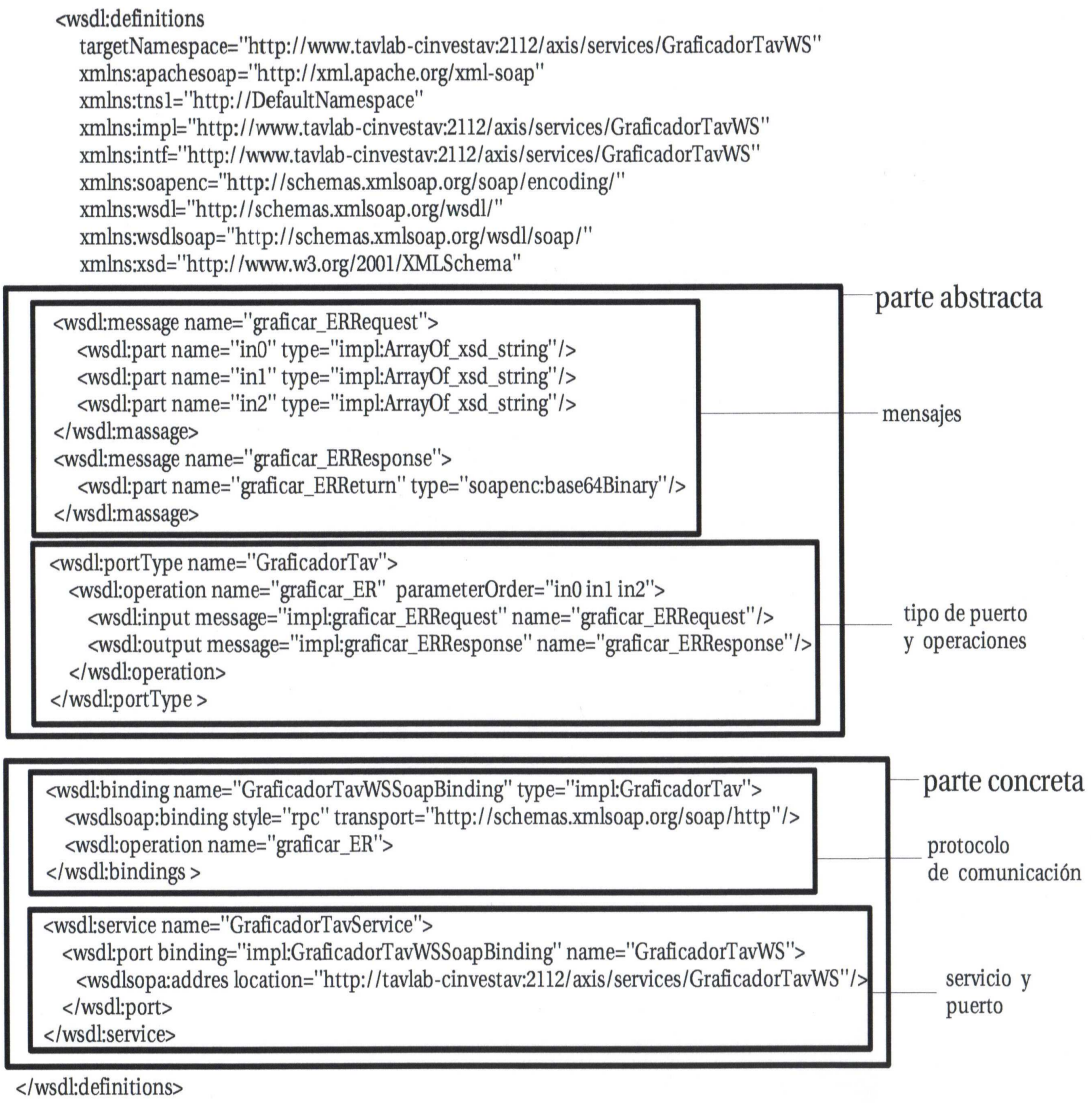


Figura 3.11: Documento WSDL correspondiente al servicio Web *GraficadorTavWS*

3.3.3 *Java proxies* para los servicios Web

Las aplicaciones cliente (*workflows*) hacen uso de los métodos expuestos por nuestros servicios Web una vez obtenidos los documentos descriptores (WSDL) y las clases Java que hacen de proxy. Un proxy es un programa proveedor de recursos localizados en máquinas remotas. Cuando un equipo conectado a Internet desea acceder a una información o recurso, es el proxy quien realiza la comunicación y a continuación traslada el resultado al equipo inicial. Los *Java proxies* se obtienen a partir de nuestros documentos WSDL mediante el comando WSDL2Java pertene-

ciente a la plataforma Apache-Axis. La *Figura 3.12* describe los pasos realizados para obtener los *Java proxies*, los cuales se explican a continuación:

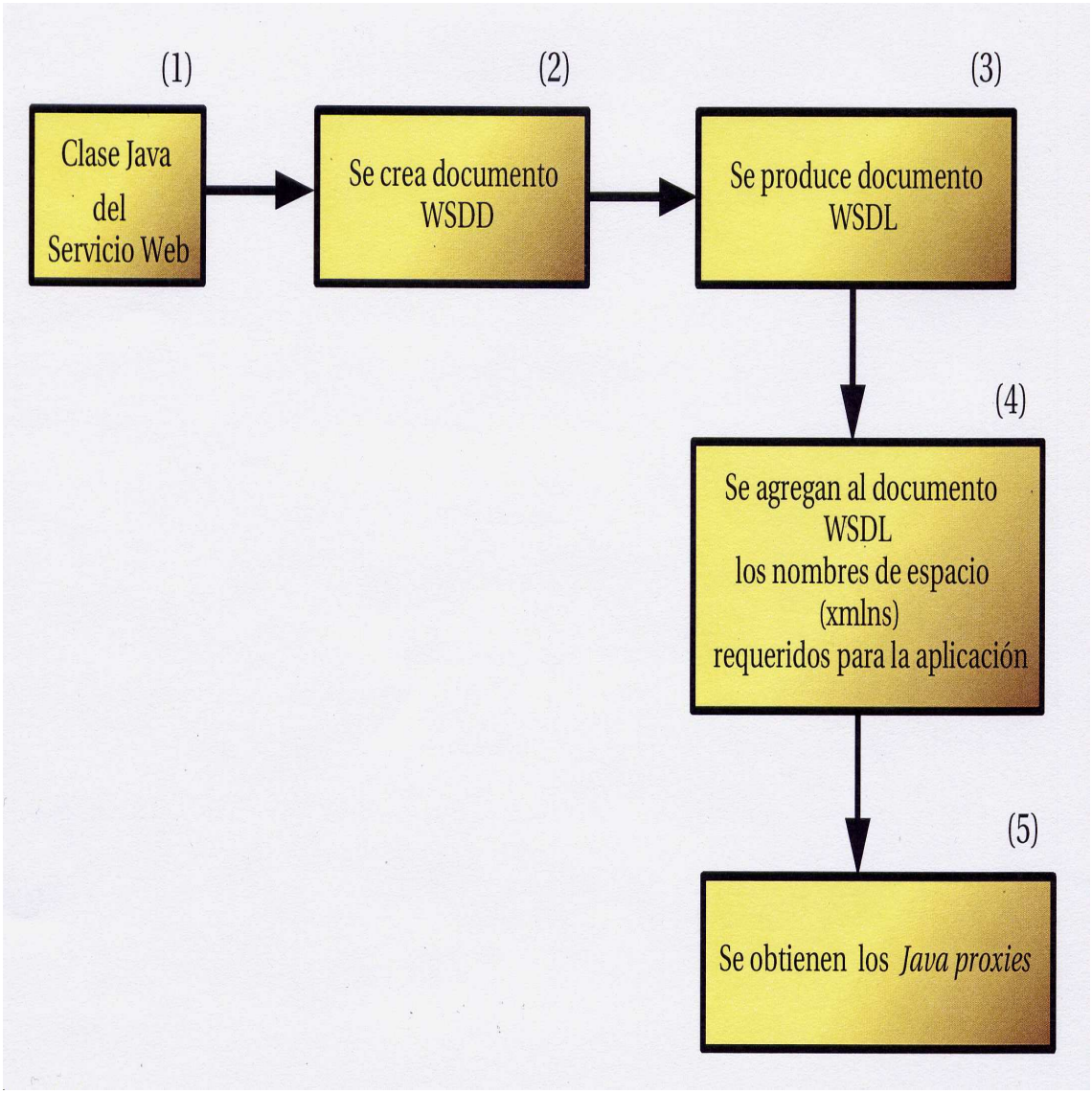


Figura 3.12: Pasos a seguir en la implementación de los servicios Web

(1) Diseñar el programa en lenguaje Java siguiendo la estructura general introducida en la Sección 3.2.1, a partir del cual se obtiene su correspondiente clase Java. Esta clase implementa al servicio Web con sus correspondientes métodos.

(2) Crear el documento WSDD para definir los métodos a ser activados. Ver Apéndice A.

(3) A partir del documento WSDD obtenemos el documento descriptor WSDL mediante la siguiente instrucción:

java org.apache.axis.client.AdminClient NombreDeLaClase.wsdd

la cual ejecuta la activación y genera un documento parcial WSDL (no incluye los nombres de espacio). En esta instrucción *NombreDeLaClase.wsdd* es el nombre del archivo correspondiente al documento WSDD, al cual se le da el mismo nombre de la clase que implementa al servicio Web con extensión *wsdd*.

(4) Mediante la revisión del documento WSDL parcial generado en el paso anterior, se determinan los nombres de espacio requeridos. Luego, se investiga en los sitios Web oficiales de la *Apache Software Foundation*, de *XML* y de *SOAP* las url's a ser seleccionadas para estos nombres de espacio.

(5) Una vez que el documento descriptor WSDL está completo, se obtienen los *Java proxies* de la aplicación mediante la siguiente instrucción:

java org.apache.axis.wsdl.WSDL2Java NombreDeLaClaseWS.wsdl

donde *NombreDeLaClaseWS.wsdl* es el nombre del archivo correspondiente al documento WSDL, al cual se le da el mismo nombre de la clase que implementa al servicio Web concatenado con las letras *WS* y con extensión *wsdl*. De manera general, después de ejecutar el comando WSDL2Java se generan los siguientes programas con extensión *java*.

- NombreDeLaClase.java
- NombreDeLaClaseService.java
- NombreDeLaClaseServiceLocator.java
- NombreDeLaClaseSoapBindingStub.java

En seguida, compilamos estos programas (mediante *javac*) para obtener así las clases correspondientes a los *Java proxies*.

De esta manera, al incluir estas clases en el mismo directorio con todas las otras clases correspondientes a nuestras aplicaciones, los recursos de servicios Web de nuestro entorno Tavlab podrán ser invocados desde cualquier sitio remoto para crear y ejecutar *workflows* científicos destinados a realizar análisis estadístico de datos dentro del ambiente de Taverna.

3.4 Iconos representativos de los métodos

En esta sección se describen los íconos representativos de los métodos *recibe_datos*, *media* y *varianza*, y se explica por medio de un ejemplo la manera de conectar las entradas y salidas para construir un *workflow* que obtenga la varianza de los datos de entrada.

La *Tabla 3.1* describe al ícono correspondiente al método *recibe_datos* perteneciente al servicio Web *RecibeDatosExperimentalesWS* y la *Tabla 3.2* describe a los métodos *media* y *varianza* correspondientes al servicio Web *ApEstadisticasWS*. En el Apéndice A se encuentran las descripciones de los íconos representativos de todos los métodos de los servicios Web implementados.

RecibeDatosExperimentalesWS

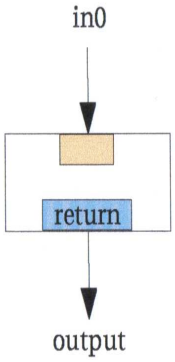
Método	Conexiones del ícono	Estructura de datos
recibe_datos		Entrada **in0**: arreglo de argumentos tipo string Salida Arreglo tipo string de datos sin líneas de retorno ni espacios

Tabla 3.1: Entrada y salida de datos del método *recibe_datos*

ApEstadisticasWS

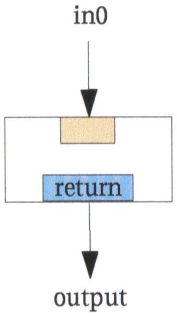
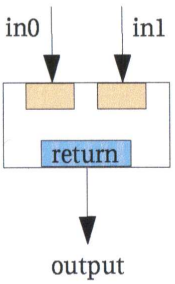
Método	Conexiones del ícono	Estructura de datos
media		Entrada **in0**: arreglo de argumentos tipo string Salida Dato tipo double
varianza		Entrada **in0**: arreglo de argumentos tipo String **in1**: dato tipo double Salida Dato tipo double

Tabla 3.2: Entrada y salida de datos de los métodos *media* y *varianza*

El diagrama de la *Figura 3.13* muestra las conexiones de las entradas y salidas de estos íconos. El ícono del método *recibe_datos* acepta un arreglo tipo string de datos en **in0**, el cual contiene, además de los mismos datos, tanto espacios ($\backslash\backslash s$) como saltos de línea ($\backslash\backslash r$) provenientes del archivo de entrada. Este método se encarga de eliminar los espacios y saltos de línea y de entregar a su salida un arreglo tipo string con sólo datos numéricos. Luego, la salida de *recibe_datos* se conecta a las entradas **in0** tanto del ícono para el método *media* como del ícono para el método *varianza*. El método *media* calcula la media de las valores introducidos y la presenta a su salida como un dato tipo *double*. Después, la salida del método *media* se conecta a la entrada **in1** del ícono para el método *varianza*. De esta manera, el método *varianza* obtiene a partir de los datos de entrada y el valor de la media introducidos en las entradas **in0** y **in1** respectivamente el valor de la varianza. Para construir un *workflow*, el usurio debe referirse al Apéndice A conteniendo las descripciones de los íconos representativos de todos los métodos implementados. La *Figura 3.14* presenta el *workflow* construido.

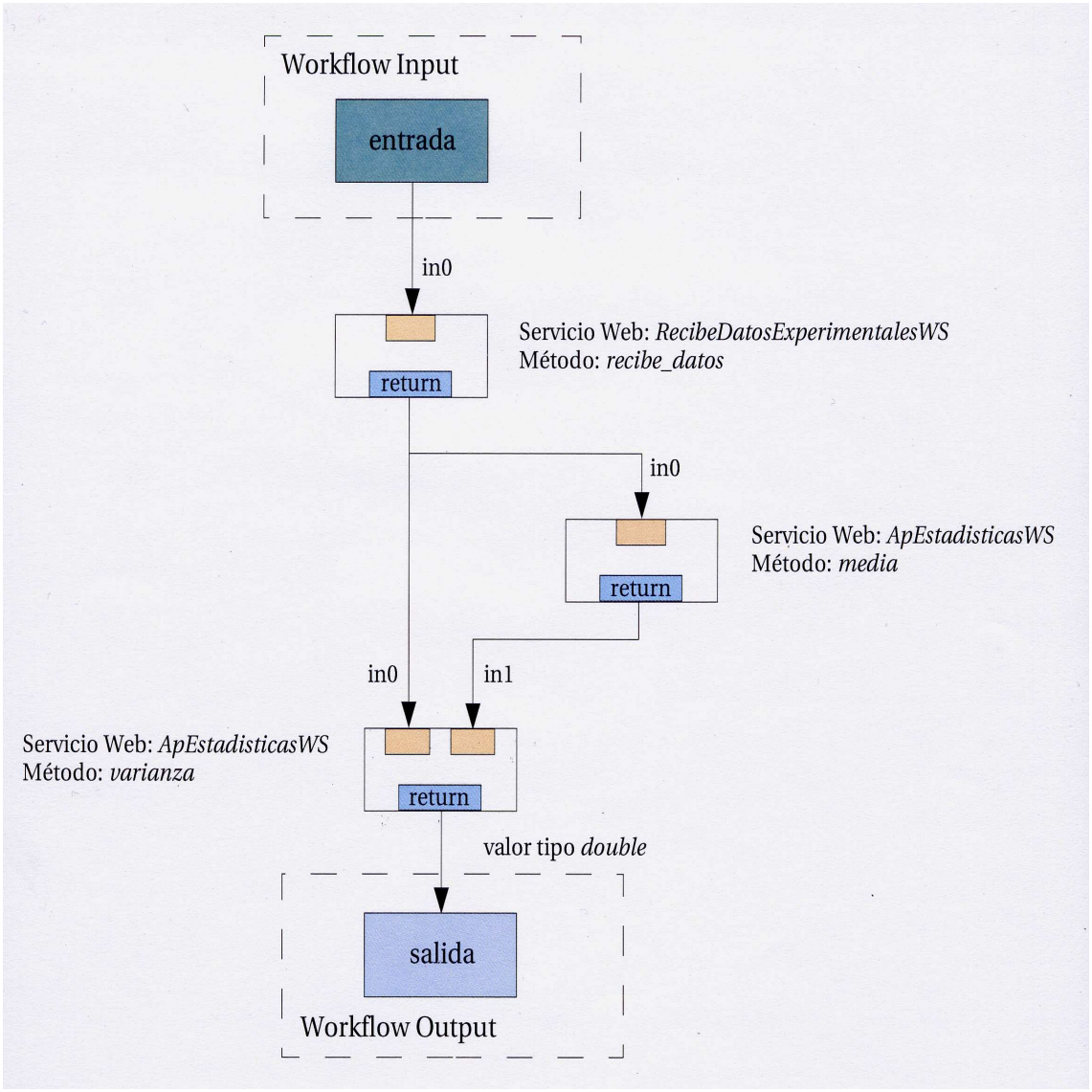


Figura 3.13: Conexiones de las entradas y salidas de los métodos

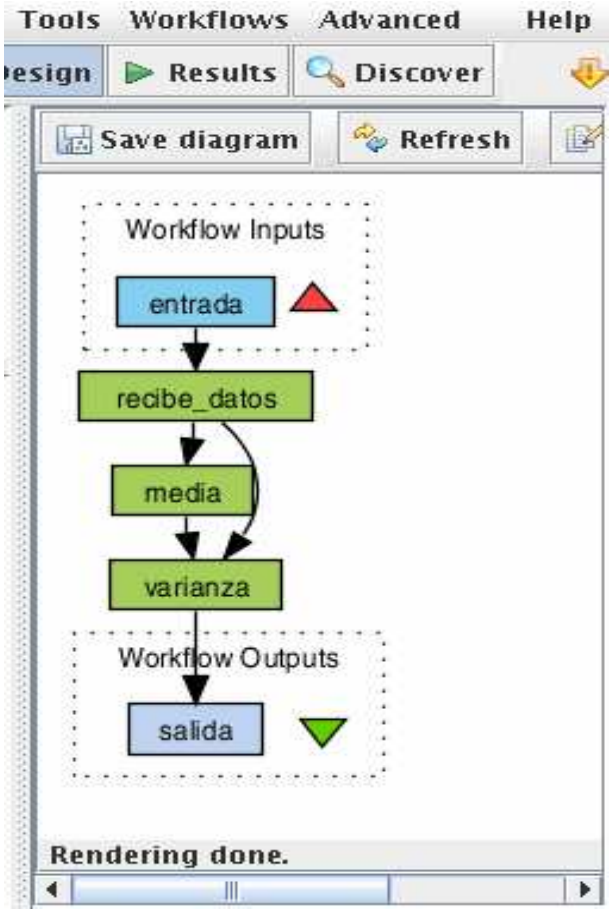


Figura 3.14: *Workflow* para obtener la varianza

3.5 Sumario

Tavlab es un entorno con recursos de servicios Web para construir *workflows* orientados a realizan análisis estadístico de datos experimentales dentro del ambiente gráfico Taverna. Los servicios Web de Tavlab diseñados y construidos como parte de nuestro trabajo, están organizados en cuatro niveles de aplicación. El nivel destinado a la administración de datos de entrada consiste de 2 servicios Web con 12 métodos para leer archivos de datos sin importar su disposición y formato (ascii o binario), y hacerlos disponibles de manera adecuada a las entradas de los *workflows* de análisis estadístico. El nivel de aplicaciones consiste de un servicio Web con 12 métodos mediante los cuales es posible realizar diferentes tipos de análisis propios de la Estadística Descriptiva como son: obtener las medidas de tendencia central y de dispersión, la ecuación de regresión, el coeficiente de correlación, aplicar la prueba t de Student, etc. Así también, consiste de un servicio Web con 6 métodos para realizar el análisis de varianza de datos experimentales y aplicar las pruebas de Tukey

y de Fisher a pares de medias. El nivel de apoyo y consulta cuenta con un servicio Web con 3 métodos para realizar cálculos aritméticos y un servicio Web con 2 métodos para realizar consultas a tablas de estadísticos necesarias en el análisis de varianza. El último nivel, presentación de resultados, consiste de 2 servicios Web con 3 métodos destinados a graficar la línea de regresión y la dispersión de los datos de entrada, así como las tablas de resultados del análisis de varianza. Debe seguirse la estructura general descrita para los programas escritos en lenguaje Java de las aplicaciones a partir de los cuales se construyen los servicios Web de nuestro ambiente Tavlab. Por último, deben obtenerse los *Java proxies* necesarios para que nuestros servicios puedan ser accesibles desde cualquier sitio remoto a través de Internet.

Capítulo 4

Aplicación de Tavlab en Biotecnología

En este capítulo describimos cómo aplicar Tavlab[1] a la resolución de problemas de análisis de varianza en Biotecnología. La Biotecnología se define como el campo del conocimiento que integra el uso de las ciencias de la Bioquímica y la Microbiología con la ingeniería para llevar a cabo aplicaciones controladas y deliberadas de las capacidades de agentes biológicos (microorganismos, cultivos celulares, etc) para uso industrial [23]. La Biotecnología tiene aplicaciones en importantes áreas industriales como son la atención a la salud, con el desarrollo de nuevos enfoques para el tratamiento de enfermedades; la agricultura con el desarrollo de cultivos y alimentos mejorados; usos no alimentarios de los cultivos, como por ejemplo plásticos biodegradables, aceites vegetales y biocombustibles; y cuidado del medio ambiente a través de la bioremediación, como el reciclaje, el tratamiento de residuos y la limpieza de sitios contaminados por actividades industriales [24]. En la Biotecnología son muchas las consideraciones que deben tomarse en cuenta al planear y ejecutar experimentos controlados. Estas consideraciones comprenden un campo de estudio en Estadística conocido como diseño de experimentos [25]. Mediante el diseño de experimentos se formula una hipótesis estadística acerca de la igualdad de los valores promedio de un conjunto de resultados experimentales obtenidos al modificar la consistencia o ambiente inicial (tratamiento) de alguna variable (parámetro), como por ejemplo, determinar si el tiempo de vida de una bacteria se ve alterado cuando se incrementa la temperatura de su ambiente en una cantidad controlada. Esta hipótesis estadística es la llamada hipótesis nula (H_0), la cual es una afirmación o conjetura acerca de los valores promedio de uno o más parámetros, y no se rechaza a menos que los resultados muestren que es falsa. La hipótesis alternativa (H_1), es una afirmación que contradice a la hipótesis nula. H_1 es aceptada si la hipótesis nula es rechazada [5]. Esto se expresa de la manera siguiente (para un solo tratamiento):

Cuando el valor promedio del parámetro analizado antes del tratamiento no se altera entonces corresponde a la hipótesis nula:

$$H_0 : \mu_1 = \mu_2 \tag{4.1}$$

[1]Entorno estadístico basado en *workflows* operado sobre la plataforma Taverna

donde:

μ_1: es el valor promedio del parámetro antes del tratamiento
μ_2: es el valor promedio del parámetro después del tratamiento

Por el contrario, cuando después de un tratamiento el valor promedio del parámetro analizado cambia, entonces corresponde a la hipótesis alternativa:

$$H_1 : \mu_1 \neq \mu_2 \tag{4.2}$$

donde:

μ_1: es el valor promedio del parámetro antes del tratamiento.
μ_2: es el valor promedio del parámetro después del tratamiento.

Para n-tratamientos en un mismo experimento la hipótesis nula (H_0) establece que el valor promedio del parámetro analizado correspondiente al i-ésimo tratamiento (μ_i), debe ser igual al valor promedio del parámetro para cualquier j-ésimo tratamiento (μ_j) con $j \neq i$:

$$H_0 : \mu_i = \mu_j, \ \ para\ toda\ i \neq j\ donde\ \ i,\ j \epsilon \{1, 2, ...n\} \tag{4.3}$$

La hipótesis alternativa (H_1), por su parte, establece que el valor promedio del parámetro analizado correspondiente al i-ésimo tratamiento (μ_i) debe ser diferente al valor promedio del parámetro para algún j-ésimo tratamiento (μ_j) con $i \neq j$:

$$H_1 : \mu_i \neq \mu_j, \ \ para\, alguna\, j \neq i,\ donde\ \ i,\ j \epsilon \{1, 2, ...n\} \tag{4.4}$$

Con el fin de aceptar o rechazar la hipótesis nula, se lleva a cabo el análisis de varianza entre pares de medias de tratamientos. Dos métodos principales para comparar pares de medias son la prueba de Tukey y la prueba de la Diferencia Significativa Mínima de Fisher (LSD, Less Significant Difference) [25]. Nuestro entorno estadístico Tavlab se utilizará en este capítulo para la resolución de problemas de análisis de varianza a través de las aplicaciones desarrolladas para los métodos de Tukey y de Fisher.

En la Sección 4.1 se describen las ecuaciones para el análisis de varianza, los cuales son: la suma de cuadrados, los grados de libertad, el cuadrado medio y el estadístico de prueba. Estas ecuaciones se utilizan en las pruebas de comparación de pares de medias de Tukey y de LSD de Fisher. En la Sección 4.2, se explica la prueba de Tukey. En la Sección 4.3, se explica la prueba de LSD de Fisher. En la Sección 4.4 se describe la organización y distribución de recursos de servicios Web de Tavlab destinados a obtener la tabla de análisis de varianza y aplicar los métodos de Tukey y de LSD de Fisher en Biotecnología. Se muestra también, que esta planificación de recursos permite añadir nuevas aplicaciones para incluir otros métodos del análisis de

varianza conforme se vayan requiriendo en el futuro. En Sección 4.5, se resuelven los ejemplos expuestos en las secciones anteriores mediante el entorno Tavlab con el fin de apreciar su utilidad en este campo.

4.1 Análisis de varianza

El análisis de varianza sirve para comparar si los valores de un conjunto de datos numéricos son significativamente distintos a los valores de otro o más conjuntos de datos. Para realizar el análisis de varianza deben conocerse los siguientes valores:

i) **Suma de cuadrados**: la suma de cuadrados total (SS_T) es una medida de la variabilidad global de los datos y está dada por:

$$SS_T = \sum_{i=1}^{a}\sum_{j=1}^{b}(y_{ji})^2 - \frac{(y)^2}{N} \tag{4.5}$$

Donde:

$\boldsymbol{a}$ es el número de tratamientos en el experimento

$\boldsymbol{b}$ es el número de réplicas en cada tratamiento

y_{ji} es la j-ésima réplica correspondiente al i-ésimo tratamiento

donde:

$i \in \boldsymbol{a}$

$j \in \boldsymbol{b}$

$\boldsymbol{N}$ = número de datos = $\boldsymbol{a}$ x $\boldsymbol{b}$

e $\boldsymbol{y}$ es la suma de todas las réplicas, esto es:

$$y = \sum_{i=1}^{b} y_j \tag{4.6}$$

La suma de cuadrados total se descompone de la forma siguiente:

$$SS_T = SS_E - SS_{Tratamientos} \tag{4.7}$$

Donde $SS_{Tratamientos}$ es la suma de cuadrados entre los tratamientos y se obtiene mediante la siguiente expresión:

$$SS_{Tratamientos} = \frac{1}{n}\sum_{i=1}^{a}(y_i)^2 - \frac{(y)^2}{N} \tag{4.8}$$

n = número de tratamientos

y SS_E es el error de la suma de cuadrados de los tratamientos, el cual se calcula mediante la diferencia:

$$SS_E = SS_T - SS_{Tratamientos} \tag{4.9}$$

ii) **Grados de libertad**: el número de grados de libertad de una suma de cuadrados es igual al número de elementos independientes en dicha suma de cuadrados. Así, los grados de libertad para cada suma de cuadrados son:

N - *1* para SS_T

$\boldsymbol{a}$ - *1* para $SS_{Tratamientos}$

N - $\boldsymbol{a}$ para SS_E

iii) **Cuadrado medio**: el cuadrado medio de los tratamientos se determina al dividir la suma de cuadrados de los tratamientos por sus respectivos grados de libertad. De igual forma, el cuadrado medio correspondiente al error se determina al dividir la suma de cuadrados del error por sus respectivos grados de libertad. Esto se muestra en las siguientes expresiones:

$$MS_{Tratamientos} = \frac{SS_{Tratamientos}}{a-1} \tag{4.10}$$

$$MS_E = \frac{SS_E}{N-a} \tag{4.11}$$

iv) **Estadístico de Prueba**: es el valor que resulta de dividir el cuadrado medio de los tratamientos por el cuadrado medio del error, esto es:

$$F_0 = \frac{MS_{Tratamientos}}{MS_E} \tag{4.12}$$

La *Tabla 4.1* de análisis de varianza resume las expresiones anteriores:

Suma de cuadrados	Grados de libertad	Cuadrado medio	F_0
$SS_{Tratamientos} = \frac{1}{n}\sum_{i=1}^{a}(y_i)^2 - \frac{(y)^2}{N}$	a-1	$MS_{Tratamientos}$	$\frac{MS_{Tratamientos}}{MS_E}$
$SS_E = SS_T - SS_{Tratamientos}$	N-a	MS_E	
$SS_T = \sum_{i=1}^{a}\sum_{j=1}^{b}(y_{ji})^2 - \frac{(y)^2}{N}$	N-1		

Tabla 4.1: Análisis de varianza

Por otro lado, en el ánalisis de varianza deben considerarse dos tipos de errores relacionados con el hecho de aceptar o rechazar la hipótesis nula. Estos son el error tipo I y el error tipo II, los cuales están representados por los valores de los parámetros α y β respectivamente. El error tipo I ocurre cuando el experimentador rechaza una hipótesis nula, cuando ésta es verdadera. El error tipo II ocurre cuando se acepta la hipótesis nula, cuando ésta es falsa. Los dos tipos de errores están inversamente relacionados, de manera que siempre se busca un balence entre α y β. En general, dicho balance se considera óptimo para un valor de $\alpha \leq 0.05$ [26].

Se presenta a continuación el siguiente ejemplo:

Un ingeniero de desarrollo de productos está interesado en saber si el peso porcentual del algodón en una fibra sintética afecta la resistencia a la tensión, por lo que ha llevado a cabo un experimento con 5 niveles (tratamientos) del peso porcentual del algodón y cinco réplicas. Los datos que obtuvo se muestran en la *Tabla 4.2* [25].

Se usará el análisis de varianza para probar la hipótesis nula:

H_0: *"los diferentes tratamientos no afectan la resistencia a la tensión de la fibra sintética"*

que se representa de acuerdo a la ec. (4.3):

$$H_0 : \mu_1 = \mu_2 = \mu_3 = \mu_4 = \mu_5$$

Contra la hipótesis alternativa:

H_1: *"algunos valores de medias son diferentes"*

que representamos de acuerdo con la ec. (4.4):

$$H_1 : \mu_i \neq \mu_j, \ \textit{para alguna } i \neq j \ \textit{donde } i, j \in \{1, 2, 3, 4, 5\} \ \textit{tratamientos}$$

Como puede observarse en la *Tabla 4.2*, la resistencia a la tensión se incrementa cuando el contenido de algodón se incrementa, hasta cerca de 30 % de algodón. Después de 30 % de algodón hay un marcado descenso de la resistencia a la tensión, es decir al rededor del 30 % de algodón produce la resistecia máxima. Con base en este análisis se tiene la firme sospecha de que el contenido de algodón afecta la resistencia a la tensión. Esto requiere de un análisis objetivo de los datos para probar la igualdad del valor de las medias. El procedimiento correcto es el análisis de varianza. De esta manera, realizamos el análisis de varianza para los datos de la *Tabla 4.2*.

Peso porcentual del algodón	Resistencia a la tensión observada (lb/pulg^2)					Totales y_i	Promedios $\bar{y}_i$
	1	2	3	4	5		
15	7	7	15	11	9	49	9.8
20	12	17	12	18	18	77	15.4
25	14	18	18	19	19	88	17.6
30	19	25	22	19	23	108	21.6
35	7	10	11	15	11	54	10.8
						y_i = 376	$\bar{y}_i$ = 15.04

Tabla 4.2: Resultados experimentales

Las sumas de cuadrados requeridas se obtienen aplicando las ecuaciones:

$$SS_T = \sum_{i=1}^{5}\sum_{j=1}^{5}(y_{ij})^2 - \frac{(y)^2}{N} \tag{4.5}$$

$$= (7)^2 + (7)^2 + (15)^2 + ... + (15)^2 + (11)^2 - \frac{(376)^2}{25} = 636.96$$

$$SS_{Tratamientos} = \frac{1}{5}\sum_{i=1}^{5}(y_i)^2 - \frac{(y)^2}{N} \tag{4.8}$$

$$= \frac{1}{5}[(49)^2 + ... + (54)^2] - \frac{(376)^2}{25} = 475.76$$

$$SS_E = SS_T - SS_{Tratamientos} \tag{4.9}$$

$$= 636.96 - 475.76 = 161.20$$

$$MS_{Tratamientos} = \frac{SS_{Tratamientos}}{\mathbf{a}-1} \tag{4.10}$$

$$= \frac{475.76}{4} = 118.94$$

$$MS_E = \frac{SS_E}{N-\mathbf{a}} \tag{4.11}$$

$$= \frac{161.20}{20} = 8.06$$

$$F_0 = \frac{MS_{Tratamientos}}{MS_E} \tag{4.12}$$

$$= \frac{118.94}{8.06} = 14.76$$

La *Tabla 4.3* resume este análisis de varianza. Comparando el valor del estadístico de prueba F_0 (14.76) con el valor que se obtiene de la tabla de puntos porcentuales de la distribución F para $\alpha = 0.05$ (incluida en el Apéndice B) para 4 y 20 grados de libertad, encontramos que $F_0 = 14.76 > 2.87$, por lo que se rechaza H_0 y se concluye que las medias de los tratamientos difieren; es decir, el peso porcentual del algodón en la fibra afecta de manera significativa la resistencia a la tensión media. En la sección 4.5 se obtendrá la misma *Tabla 4.3* de análisis de varianza utilizando nuestro entorno gráfico Tavlab.

Suma de cuadrados	Grados de libertad	Cuadrado medio	F_0
$SS_{Tratamientos} = 475.76$	4	118.94	14.76
$SS_E = 161.20$	20	8.06	
$SS_T = 636.96$	24		

Tabla 4.3: Resultados del análisis de varianza

Por otra parte, el análisis de varianza requiere en ocasiones comparar sólo pares de medias. Frecuentemente, es posible determinar cuáles son las medias que difieren probando las diferencias entre todos los pares de medias de los tratamientos. La prueba de Tukey y del LSD de Fisher se aplican específicamente para realizar las comparaciones por pares entre todas las medias poblacionales.

4.2 La prueba de Tukey

La prueba de Tukey [25] consiste en determinar un valor de referencia respecto al cual las diferencias de las medias son comparadas. Si la diferencia para un par de medias es menor que este valor, se considera como no significativa.

Para calcular el valor de referencia de Tukey se aplica la siguiente fórmula:

$$Tukey = q_\alpha(\mathbf{a}, f)\sqrt{\frac{MS_E}{n}} \tag{4.13}$$

Donde MS_E es el cuadrado medio debido al error (ec. 4.11), n es el número de tratamientos y $q_\alpha(\mathbf{a}, f)$ es un factor que se obtiene de la tabla para la distribución del estadístico de rango studentizado q con $\alpha = 0.05$ (incluida en el Apéndice B) a partir del número de tratamientos $\mathbf{a}$ y los grados de libertad f asociados con MS_E (ec. 4.11).

A continuación, se aplica la prueba de Tukey (ec. 4.13) para encontrar las diferencias que sean significativas entre todas las medias del problema de la Sección 4.1. Para esto, se utilizan los valores de la *Tabla 4.3* de análisis de varianza obtenidos anteriormente:

a) Los grados de libertad:

$$N - \mathbf{a} = 25 - 5 = 20$$

b) Las sumas de cuadrados:

$$SS_{Tratamientos} = 475.76 \tag{4.8}$$

$$SS_T = 636.96 \tag{4.5}$$

$$SS_E = 161.20 \tag{4.9}$$

c) Cuadrado medio debido al error:

$$MS_E = 8.06 \tag{4.11}$$

d) Valor del factor $q_{0.05}(5, 20)$:

$$q_{0.05}(5, 20) = 4.23$$

De esta manera, el valor de referencia de Tukey resulta:

$$Tukey = q(\mathbf{a}, f)\sqrt{\tfrac{MS_E}{n}} \tag{4.13}$$

$$= 4.23\sqrt{\tfrac{8.06}{5}} = 5.37$$

Por lo tanto, cualquier par de medias de los tratamientos cuya difierencia en valor absoluto sea mayor de **5.37** implicaría que el par correspondiente de medias poblacionales son significativamente diferentes. Los cinco promedios de los tratamientos son:

$$\tilde{y_1} = 9.8 \quad \tilde{y_2} = 15.4 \quad \tilde{y_3} = 17.6 \quad \tilde{y_4} = 21.6 \quad \tilde{y_5} = 10.8$$

y las diferencias en los promedios son:

$\tilde{y_1} - \tilde{y_2} = 9.8 - 15.4 = -5.6^d$
$\tilde{y_1} - \tilde{y_3} = 9.8 - 17.6 = -7.8^d$
$\tilde{y_1} - \tilde{y_4} = 9.8 - 21.6 = -11.8^d$
$\tilde{y_1} - \tilde{y_5} = 9.8 - 10.8 = -1.0$
$\tilde{y_2} - \tilde{y_3} = 15.4 - 17.6 = -2.2$
$\tilde{y_2} - \tilde{y_4} = 15.4 - 21.6 = -6.2^d$
$\tilde{y_2} - \tilde{y_5} = 15.4 - 10.8 = 4.6$
$\tilde{y_3} - \tilde{y_4} = 17.6 - 21.6 = -4.0$
$\tilde{y_3} - \tilde{y_5} = 17.6 - 10.8 = 6.8^d$
$\tilde{y_4} - \tilde{y_5} = 21.6 - 10.8 = 10.8^d$

Así, mediante la prueba de Tukey, se han encontrado los contrastes de la forma $\Gamma = \mu_i - \mu_j$ para toda $i \neq j$ que difieren significativamente (etiquetados con la letra d).

Por lo tanto, la hipótesis nula se rechaza:

H_0: *"los diferentes tratamientos no afectan la resistencia a la tensión de la fibra sintética"*

y se acepta la hipótesis alternativa:

H_1: *"algunos valores de medias son diferentes"*

4.3 La prueba LSD (*Less Significant Difference*) de Fisher

La prueba de Fisher [25] consiste en determinar el valor de la *diferencia significativa mínina* (LSD) con respecto al cual comparar las diferencias de las medias de los tratamientos. Si la diferencia para un par de medias es menor que este valor, se considera como no significativa. La *diferencia significativa mínima* (LSD), se calcula mediante la siguiente expresión:

$$LSD = t_{\alpha/2,\, N-a}\sqrt{\frac{2MS_E}{n}} \tag{4.14}$$

Donde $t_{\alpha/2,\, N-a}$ es el valor de la distribución t para α puntos porcentuales y N - **a** grados de libertad (incluida en el Apéndice B). MS_E es el cuadrado medio del error (ec. 4.11) y n es el número de réplicas. Para usar el procedimiento LSD de Fisher, simplemente se compara la diferencia observada entre cada par de promedios con la LSD correspondiente. Para ilustrar el procedimiento, se presenta el siguiente ejemplo:

Si se usan los datos del experimento presentado anteriormente acerca de la resistencia a la tensión de una fibra sintética con respecto al peso porcentual del algodón, la LSD con $\alpha = 0.05$ y $t_{0.25,\,20} = 2.086$ es:

$$LSD = t_{\alpha/2,\, N-a}\sqrt{\frac{2MS_E}{n}} = 2.086\sqrt{\frac{2(8.06)}{5}} = 3.75 \tag{4.14}$$

De esta manera, cualquier par de promedios de los tratamientos que difiera del valor absoluto por más de **3.75** implicaría que el par correspondiente de medias poblacionales es significativamente diferente. Las diferencias de los promedios son:

$\tilde{y_1} - \tilde{y_2} = 9.8 - 15.4 = -5.6^{lsd}$
$\tilde{y_1} - \tilde{y_3} = 9.8 - 17.6 = -7.8^{lsd}$
$\tilde{y_1} - \tilde{y_4} = 9.8 - 21.6 = -11.8^{lsd}$
$\tilde{y_1} - \tilde{y_5} = 9.8 - 10.8 = -1.0$
$\tilde{y_2} - \tilde{y_3} = 15.4 - 17.6 = -2.2$
$\tilde{y_2} - \tilde{y_4} = 15.4 - 21.6 = -6.2^{lsd}$
$\tilde{y_2} - \tilde{y_5} = 15.4 - 10.8 = 4.6^{lsd}$
$\tilde{y_3} - \tilde{y_4} = 17.6 - 21.6 = -4.0^{lsd}$
$\tilde{y_3} - \tilde{y_5} = 17.6 - 10.8 = 6.8^{lsd}$
$\tilde{y_4} - \tilde{y_5} = 21.6 - 10.8 = 10.8^{lsd}$

Así también, mediante la prueba del LSD de Fisher, se han encontrado los contrastes de la forma $\Gamma = \mu_i - \mu_j$ para toda $i \neq j$ que difieren significativamente (etiquetados con lsd).

Por lo tanto, esta prueba también conduce a rechazar la hipótesis nula:

H_0: *"los diferentes tratamientos no afectan la resistencia a la tensión de la fibra sintética"*

y se acepta la hipótesis alternativa:

H_1: *"algunos valores de medias son diferentes"*

Sin embargo, mediante la prueba de LSD de Fisher sólo dos contrastes no difieren significativamente en comparación con la prueba de Tukey para la cual se encontraron cuatro contrastes sin ninguna diferencia significativa. Esto se debe a que la prueba de Fisher es más sensible a las diferencias existentes, manteniendo bajo el error tipo I (rechazar la hipótesis nula cuando ésta es verdadera).

4.4 Organización y distribución de recursos de servicios Web

En esta sección veremos cómo están organizados y distribuidos los diferentes recursos de servicios Web de Tavlab destinados a obtener la tabla de análisis de varianza y aplicar las pruebas de Tukey y Fisher en Biotecnología. Se mostrará también, cómo esta planificación de recursos permitirá fácilmente añadir nuevas aplicaciones para incluir otros métodos del análisis de varianza conforme se vayan requiriendo en el futuro y hacer nuestro entorno de resolución de problemas distribuidos más óptimo.

Los recursos de servicios Web de nuestro entorno Tavlab diseñados e implementados para obtener la tabla de análisis de varianza, así como para aplicar las pruebas de Tukey y de LSD de Fisher en el análisis de experimentos en Biotecnología, se muestran en la *Figura 4.1.* En la *Figura 4.1* podemos apreciar que estos recursos consisten principalmente de cuatro servicios Web con varios métodos para llevar a cabo los diferentes análisis estadísticos. Estos servicios están disponibles en el sitio Web para Tavlab *http://www.tavlab-cinvestav.mx:2112/* y descritos en el capítulo 3:

- El Servicio Web (*SW1*) *RecibeDatosExperimentalesWS* cuenta con métodos diseñados para recibir datos de entrada dependiendo de la disposición de éstos en el archivo con que se alimentará el *workflow.*
- El Servicio Web (*SW2*) *AnalisisDeVarianzaWS* obtiene cada uno de los elementos de la tabla de análisis de varianza (*Tabla 5.1*), así también, suministra los valores requeridos por los métodos de las pruebas estadísticas de Tukey y Fisher.
- En el Servicio Web (*SW3*) *TablasDeEstadisticosWS* los métodos implementados corresponden a las tablas de los estadísticos $\boldsymbol{q}$ y $\boldsymbol{t}$, incluidas en el Apéndice B. Los valores de estos estadísticos son necesarios para las pruebas estadísticas de Tukey y de LSD de Fisher. Cuando se requiera incluir más tablas de otros estadísticos, los métodos correspondientes simplemente deberán adicionarse a este mismo servicio Web.

- El Servicio Web (*SW4*) *PruebasEstadisticasWS* cuenta con los métodos diseñados para aplicar las pruebas de Tukey y de LSD de Fisher. Así también, cuando se requiera implementar otras pruebas estadísticas para el análisis de experimentos, podrán ser incluidas en este mismo servicio Web.

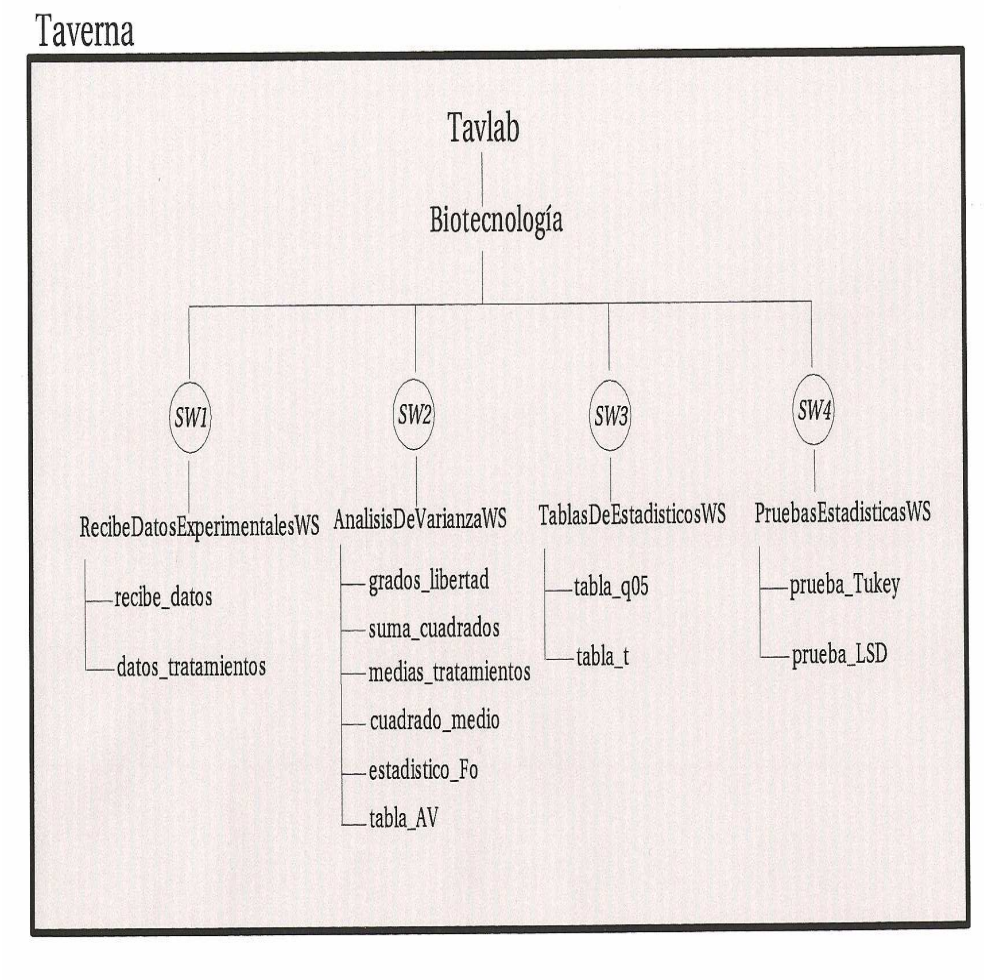


Figura 4.1: Organización de los recursos de servicios Web empleados en Biotecnología

En la *Figura 4.2* se desglozan los métodos de cada uno de estos servicios Web utilizados en las tres aplicaciones: tabla de análisis de varianza, prueba de Tukey y la prueba LSD de Fisher. Por ejemplo, la *Figura 4.2* muestra que para realizar el análisis de varianza se requiere el método diseñado para recibir los datos experimentales del servicio Web *SW1*, así como los seis métodos correspondientes al servicio Web *SW2*. Así también, para realizar tanto la prueba de Tukey como la de LSD de Fisher, se requiere el método destinado a recibir datos experimentales del servicio Web *SW1*, así como los métodos dedicados a calcular los grados de libertad, la suma de cuadrados y las medias de los tratamientos correspondientes al servicio Web *SW2*.

Además, para la prueba de Tukey debe utilizarse el método con los valores del estadístico **q** para $\alpha = 0.05$ del servicio Web *SW3*, así como el método que obtiene la tabla con los resultados de la prueba de Tukey del servicio Web *SW4*. Asimismo, para la prueba de LSD debe utilizarse el método con los valores del estadístico **t** del servicio Web *SW3*, así como el método que obtiene la tabla con los resultados de esta prueba correspondiente al servicio Web *SW4*. En el Apéndice A, se describen las interconexiones entre estos métodos, las cuales forman un *workflow* específico en el entorno Tavlab.

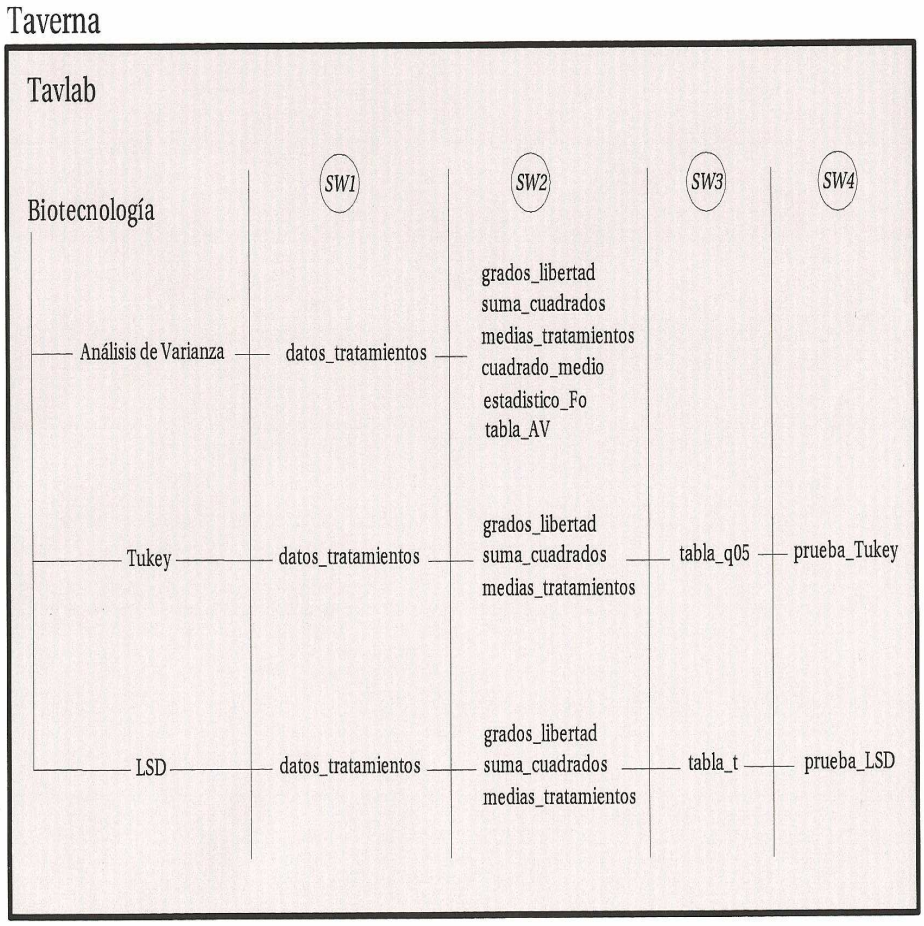


Figura 4.2: Métodos de servicios Web diseñados para las aplicaciones en Biotecnología

4.5 Utilización del entorno estadístico *Tavlab*

Con la finalidad de apreciar claramente cómo emplear nuestro entorno Tavlab, realizaremos el análisis de varianza, así como las pruebas de Tukey y de LSD de Fisher al ejemplo del peso porcentual del algodón en una fibra sintética introducido en la Sección 4.1. Como se mencionó en el Capítulo 3, se podrá ver que la utilización de nuestro entorno Tavlab aporta varios beneficios como son, construir *workflows* a partir de recursos de servicios Web distribuidos en varios sitios en Internet, así como utilizar datos experimentales localizados no sólo en el sitio de trabajo, sino también en cualquier otro sitio. Aprovechando así, una mayor capacidad de cómputo utilizando servidores distribuidos. Así también, la estructura de nuestro entorno Tavlab, permite fácilmente su crecimiento mediante la inclusión de nuevas aplicaciones y recursos conforme se requieran. Por otra parte, se omitirán los pasos para construir un *workflow*, suministrarle los datos de entrada, ejecutarlo y guardar los resultados en un archivo, ya que éstos están explicados en el Capítulo 2.

4.5.1 Obtención de la tabla de análisis de varianza

La *figura 4.3* muestra el *workflow* construido a partir de los recursos de nuestro entorno estadístico Tavlab para obtener la tabla de análisis de varianza (*Tabla 4.1*). De esta manera, tenemos:

Definición del problema: dado un conjunto de datos experimentales y un número de tratamientos determinados, crear un *workflow* para realizar el análisis de varianza.

Meta: obtener la tabla de analisis de varianza tal como se muestra en la *Tabla 4.1* a partir del método tabla_AV.

Solución: para obtener la tabla de análisis de varianza a partir del método *tabla_AV*, se deben calcular primero los grados de libertad, la suma de cuadrados y el estadístico Fo de los datos experimentales mediante los métodos *grados_libertad*, *suma_cuadrados* y *estadistico_Fo* respectivamente. A su vez, el estadístico Fo se obtiene a partir del cuadrado medio, utilizando el método *cuadrado_medio*, y éste a partir de los grados de libertad y suma de cuadrados. Luego, estos métodos recibirán los datos de entrada en forma apropiada (arreglo tipo *string* de datos) suministrados por el método *datos_tratamientos*. En la *Figura 4.4* se presenta la ventana en la que se introducen las URLs de todos estos servicios Web para que sean localizados en nuestro sitio para Tavlab:

1. http://www.tavlab-cinvestav.mx:1221/axis/
   services/RecibeDatosExperimentalesWS?wsdl
2. http://www.tavlab-cinvestav.mx:1221/axis/
   services/AnalisisDeVarianzaWS?wsdl

3. http://www.tavlab-cinvestav.mx:1221/axis/
services/TablasDeEstadisticosWS?wsdl

4. http://www.tavlab-cinvestav.mx:1221/axis/
services/PruebasEstadisticasWS?wsdl

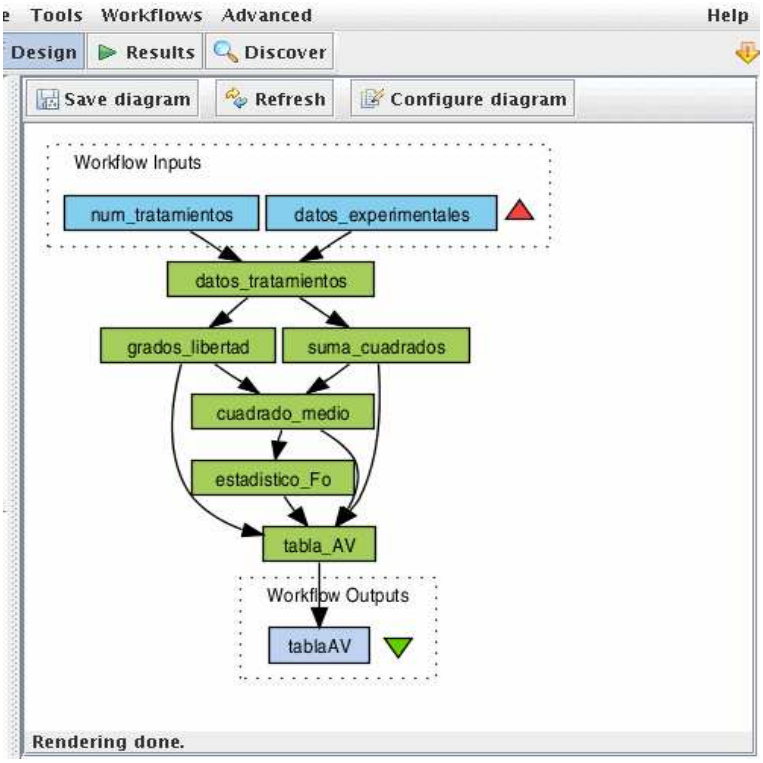


Figura 4.3: *Workflow* para obtener la tabla de análisis de varianza

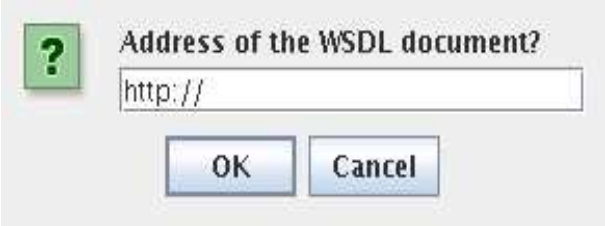


Figura 4.4: Ventana en la que se introducen las URLs de los servicios Web

De esta manera, en el panel de servicios disponibles de Taverna, aparecerán estos servicios Web con sus respectivos métodos como se muestra en la *Figura 4.5*.

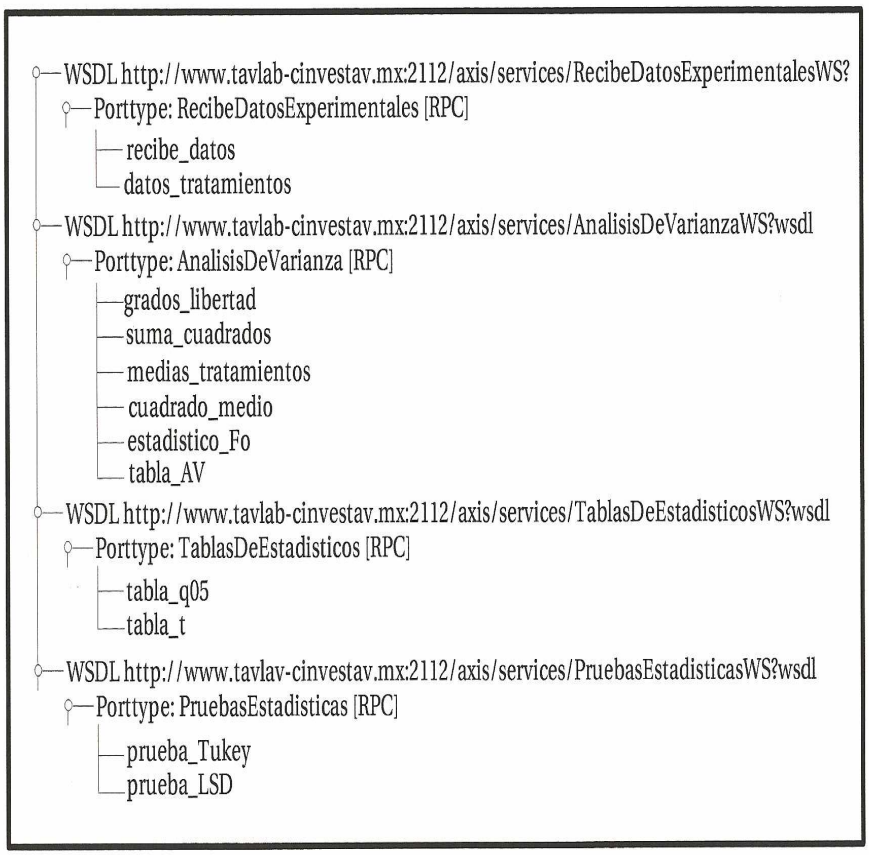


Figura 4.5: Panel de servicios disponibles en Taverna

En seguida, seleccionando los métodos correspondientes del panel de servicios, se construye el *workflow* (*Figura 4.3*) para obtener la tabla de análisis de varianza. Para ilustrar la manera de conectar los métodos, el diagrama de la *Figura 4.6* muestra las conexiones de las entradas y salidas de los íconos representativos de los métodos del *workflow* de la *Figura 4.3*. Para construir un *workflow*, el usuario debe referirse al Apéndice A conteniendo las descripciones de los íconos representativos de todos los métodos implementados. De esta manera, la *Figura 4.6* muestra el nombre del método correspondiente a cada entrada.

Los datos experimentales que se suministrarán al *workflow* son los resultados del experimento mostrados en la *Tabla 4.2*, y que se encuentran en el sitio Web: *http://www.matrix-cinvestav.mx/~urevilla/data/datos_ experimentales.txt*. Esto se muestra en la *Figura 4.7*.

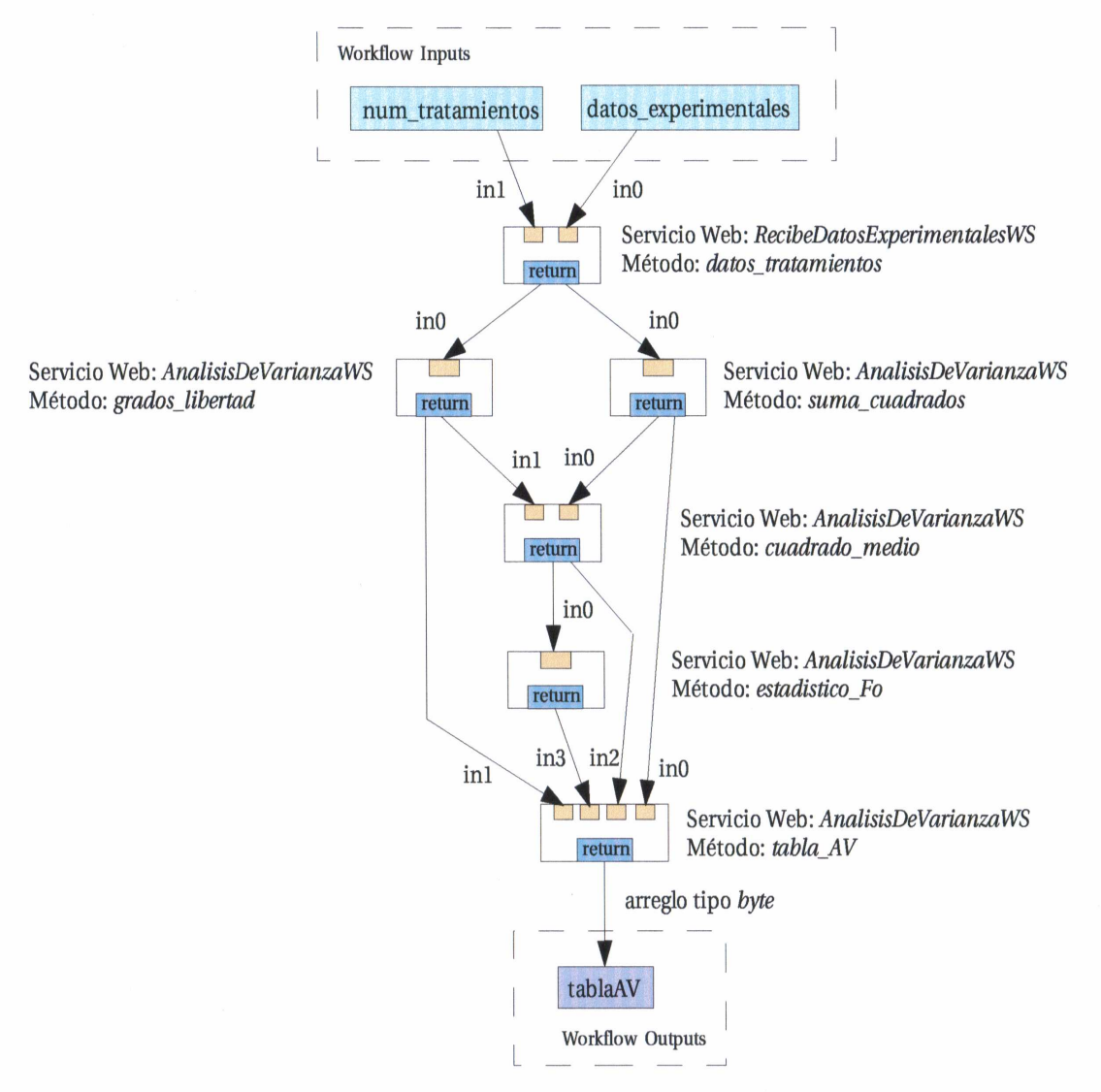


Figura 4.6: Conexiones de las entradas y salidas de los métodos

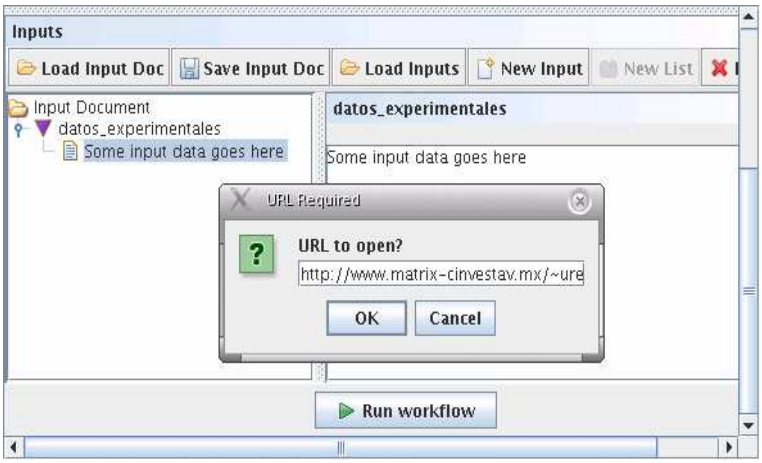


Figura 4.7: Se alimenta al *workflow* con los datos experimentales

A continuación, introducimos también el número de tratamientos (los diferentes pesos porcentuales del algodón) considerados en el experimento, como muestra la *Figura 4.8* es 5.

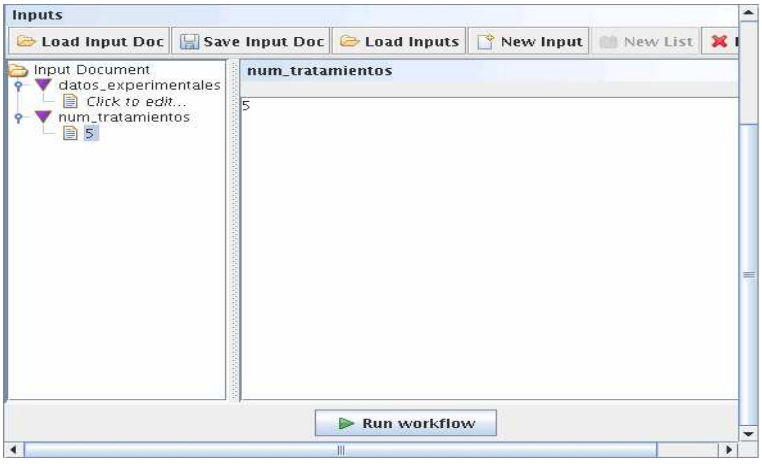


Figura 4.8: Se alimenta al *workflow* con el número de tratamientos

Así, ejecutamos el *workflow* presionando el botón *Run workflow* mostrado también en la *Figura 4.8*. Como resultado obtenemos la *Tabla 4.4* correspondiente a la tabla de análisis de varianza.

Tabla de Análisis de Varianza				
Suma de cuadrados		Grados de libertad	Cuadrado medio	Fo
SS_Tratamientos	475.76	4	118.94	14.76
SS_E	161.20	20	8.06	
SS_T	636.96	24		

Tabla 4.4: Análisis de varianza obtenida utilizando el entorno Tavlab

4.5.2 Aplicación de la prueba LSD de Fisher

Para aplicar la prueba de LSD de Fisher descrita en la Sección 4.3 a los datos experimentales mediante nuestro entorno Tavlab, se considera la misma disposición de recursos de servicios Web y se construye el *workflow* de la *Figura 4.8*. De esta manera, tenemos:

Definición del problema: dado un conjunto de datos experimentales y un número de tratamientos determinados, crear un *workflow* para aplicar la prueba LSD de Fisher.

Meta: obtener una tabla que muestre las *diferencias significativas mínimas* entre los pares de medias de los tratamientos.

Solución: para obtener esta tabla con las *diferencias significativas mínimas* se deben calcular primero las medias de los tratamientos, los grados de libertad y la suma de cuadrados de los datos experimentales mediante los métodos: *medias_tratamientos*, *grados_libertad* y *suma_cuadrados* respectivamente. Así también, se necesita obtener el valor del estadístico **t** mediante el método *tabla_t* al suministrar el parámetro α y los grados de libertad. Luego, estos métodos recibirán los datos de entrada en forma apropiada (arreglo tipo *string* de datos) suministrados por el método *datos_tratamientos* (*Figura 4.9*).

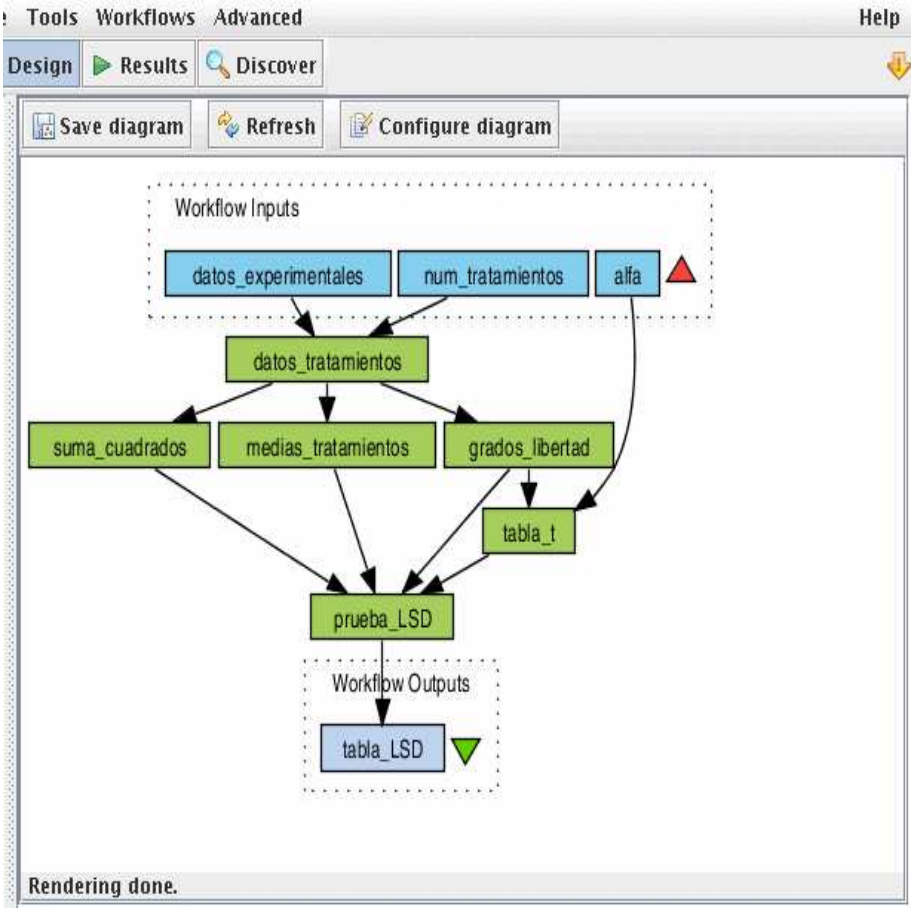


Figura 4.9: *Workflow* para obtener la tabla de la prueba del LSD

A continuación suministramos al *workflow* los datos experimentales ubicados en el sitio Web:

http://www.matrix-cinvestav.mx/~urevilla/data/datos_experimentales.txt

en la misma forma como se muestra en las *Figuras 4.7* y *4.8*. Así también, introducimos el valor del parámetro $\alpha = 0.05$ como se indica en la *Figura 4.10*.

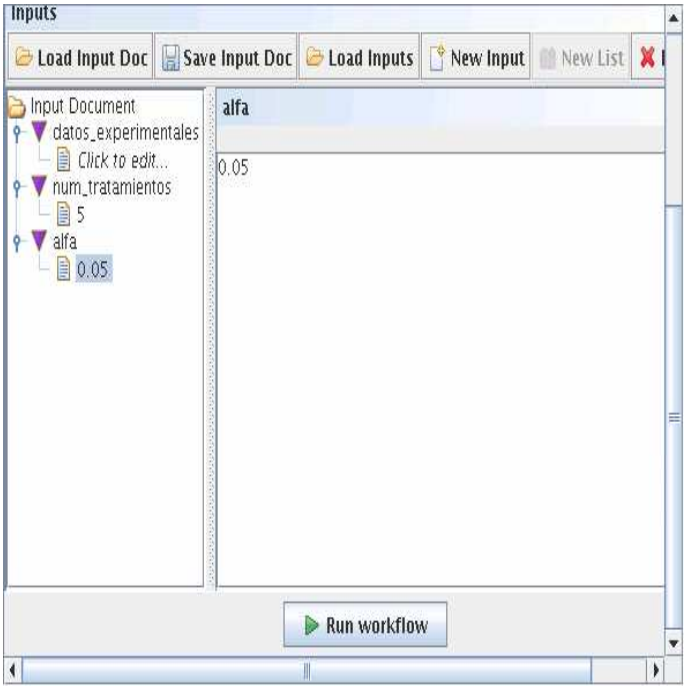


Figura 4.10: Se introduce el valor del parámetro alfa

Después de ejecutar el *workflow*, se obtiene la tabla de la *Figura 4.11* correspondiente a la prueba de LSD de Fisher.

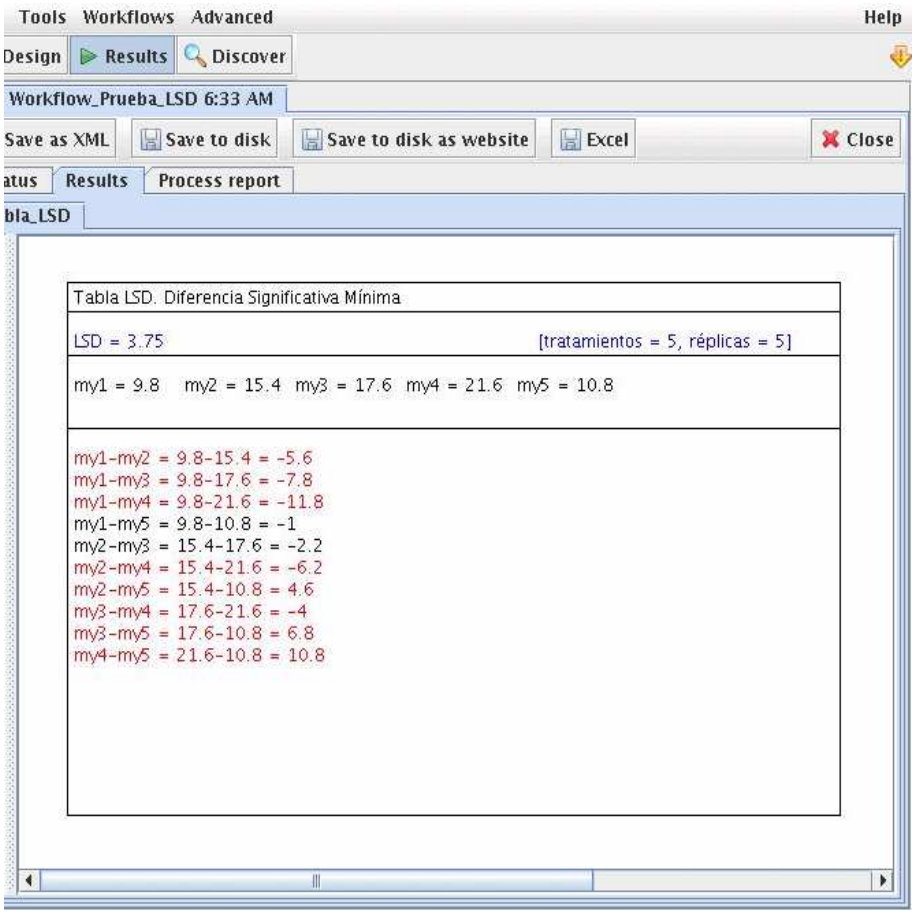


Figura 4.11: Resultados de la prueba de LSD de Fisher obtenidos con Tavlab

4.5.3 Aplicación de la prueba de Tukey

Para esta aplicación se han distribuido los recursos de servicios Web en 2 sitios, *AnalisisDeVarianzaWS* y *TablasDeEstadisticosWS* se encuentran en nuestro sitio donde está Tavlab, y tanto *RecibeDatosExperimentalesWS* como *PruebasEstadisticasWS* se instalaron en el sitio Tux. Esta disposición distribuida en diferentes sitios de nuestros recursos se muestra en la *Figura 4.12*.

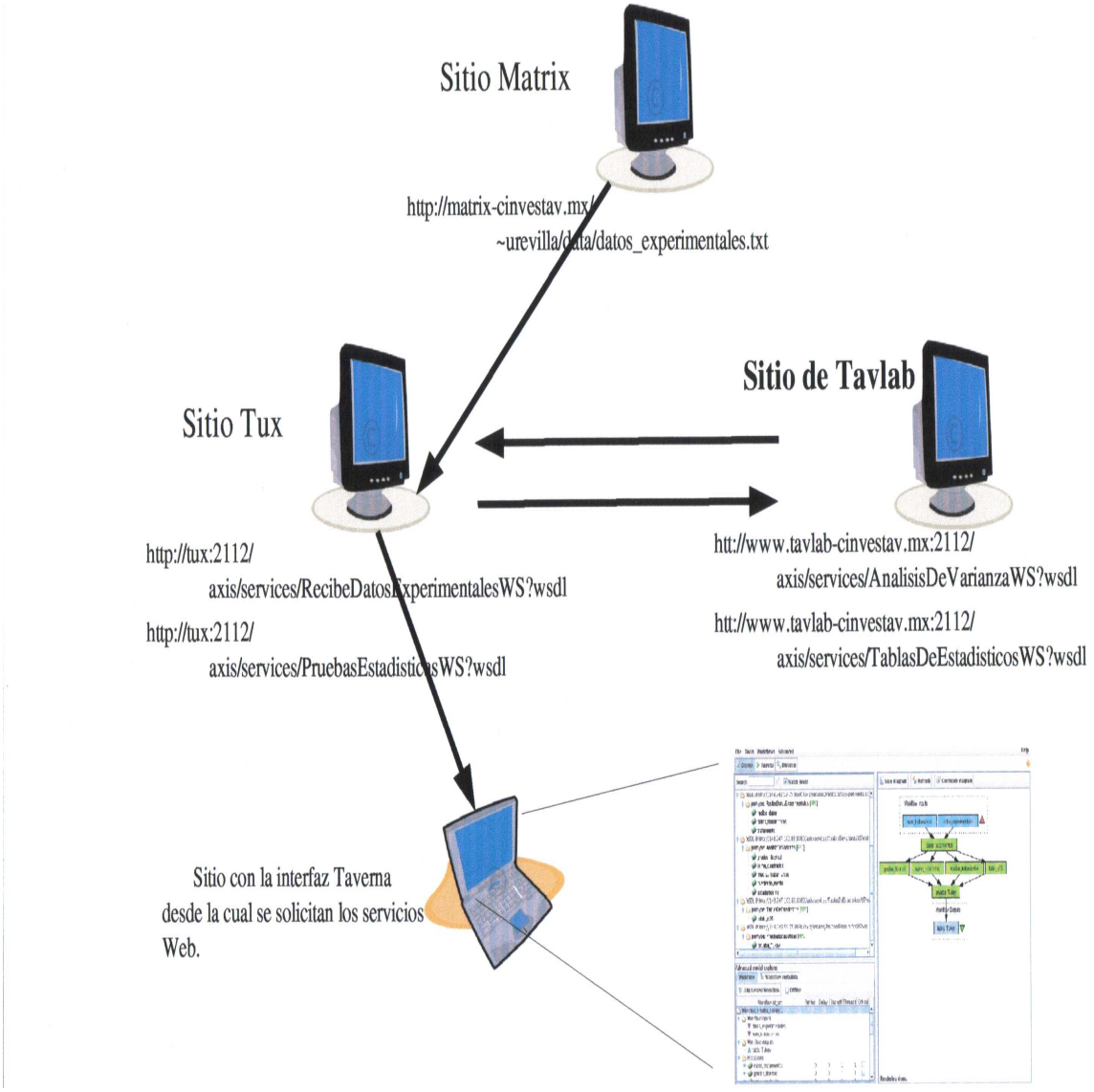


Figura 4.12: Recursos para Biotecnología de Tavlab distribuidos en 2 sitios

De esta manera, ahora nuestros servicios Web se obtienen desde estas ubicaciones:

1. http://tux:2112/axis/services/RecibeDatosExperimentalesWS?wsdl

2. http://www.tavlab-cinvestav.mx:2112/axis/
services/AnalisisDeVarianzaWS?wsdl

3. http://www.tavlab-cinvestav.mx:2112/axis/
services/TablasDeEstadisticosWS?wsdl

4. http://tux:1221/axis/services/PruebasEstadisticasWS?wsdl

Desplegándose en el panel de servicios disponibles de Taverna como muestra la *Figura 4.13*.

Figura 4.13: Panel de servicios disponibles de Taverna

La *Figura 4.14* muestra el *workflow* construido a partir de los recursos de servicios Web localizados en el sitio donde está Tavlab y en el sitio Tux.

Tenemos ahora que:

Definición del problema: dado un conjunto de datos experimentales y un número de tratamientos determinados, crear un *workflow* para aplicar la prueba de Tukey.

Meta: obtener una tabla mostrando las diferencias significativas entre los pares de medias de los tratamientos de acuerdo a la prueba de Tukey.

Solución: para obtener esta tabla con las diferencias significativas según la prueba de Tukey, se deben calcular primero las medias de los tratamientos, los grados de

libertad y la suma de cuadrados de los datos experimentales mediante los métodos *medias_tratamientos*, *grados_libertad* y *suma_cuadrados* respectivamente. Así también, se necesita obtener el valor del estadístico **q** mediante el método *tabla_q05*. Luego, estos métodos recibirán los datos de entrada en forma apropiada (arreglo tipo *string* de datos) suministrados por el método *datos_tratamientos*.

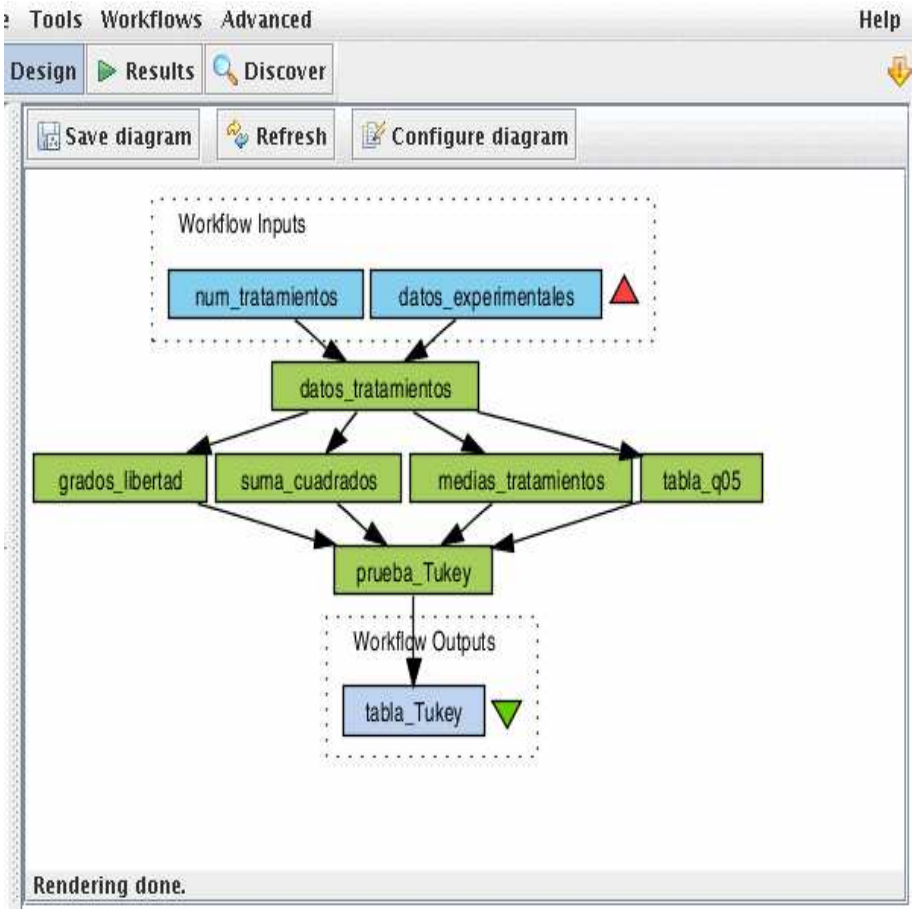


Figura 4.14: *Workflow* para realizar la prueba de Tukey

A continuación suministramos al *workflow* los datos experimentales ubicados en el sitio Web:

http://www.matrix-cinvestav.mx/~urevilla/data/datos_experimentales.txt

e introducimos el número de tratamientos como se mostró en las (*Figuras 4.7* y *4.8*).

Después de ejecutar el *workflow*, se obtiene la tabla de la *Figura 4.15* correspondiente a la prueba de Tukey.

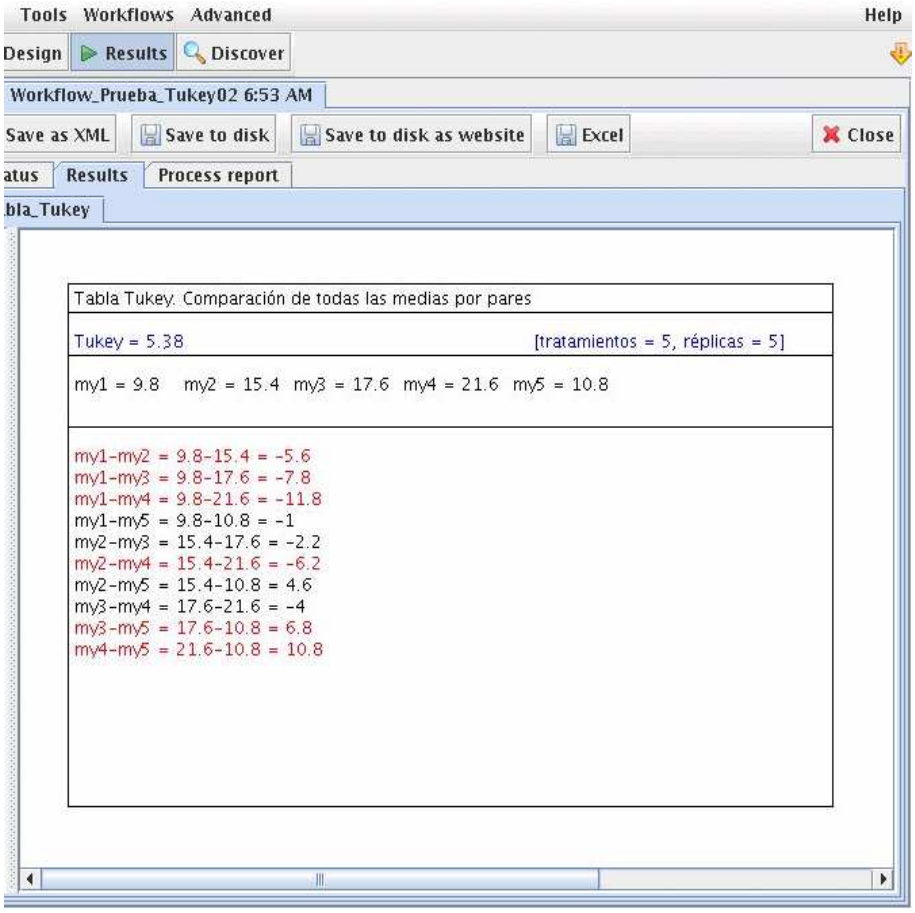


Figura 4.15: Resultados de la prueba de Tukey obtenidos con Tavlab

4.6 Sumario

En este capítulo se describieron los recursos de nuestro entorno estadístico Tavlab diseñados e implementados para aplicarse en la resolución de problemas de análisis de varianza en Biotecnología, como son: la obtención de la tabla de análisis de varianza, la cual contiene los valores para la suma de cuadrados, los grados de libertad, el cuadrado medio y el estadístico Fo correspondientes a los resultados de un experimento, y la aplicación de las pruebas de Tukey y de LSD de Fisher.

Se mostró que estos recursos consisten principalmente de cuatro servicios Web, los cuales cuentan con varios métodos para llevar a cabo los correspondientes análisis, así como de un sitio Web desde el cual se pueden acceder estas aplicaciones. Así también, se presentó un ejemplo donde nuestras aplicaciones están distribuidas en dos sitios en Internet para mostrar que los recursos de Tavlab pueden estar eventualmente distribuidos.

De esta manera, nuestro entorno Tavlab aporta varios beneficios como son, construir *workflows* a partir de recursos de servicios Web que se encuentran distribuidos

en varios sitios en Internet, así como utilizar datos experimentales que se localizan no sólo en el sitio de trabajo, sino también en cualquier otro sitio. Aprovechando así, una mayor capacidad de cómputo al utilizar servidores distribuidos. Asimismo, la estructura de nuestro entorno Tavlab permite fácilmente su crecimiento mediante la inclusión de nuevas aplicaciones y recursos que se requieran en el futuro.

Capítulo 5

Quimiometría y Medicina Estadística

En este capítulo describimos cómo aplicar Tavlab[1] a la resolución de problemas de análisis estadístico de datos experimentales en las áreas de la Quimiometría y la Medicina Estadística. La Quimiometría es la disciplina química que se enfoca en la aplicación de métodos matemáticos o estadísticos sobre datos químicos[27]. La Medicina Estadística se apoya en métodos estadísticos para la investigación de los mecanismos de la enfermedad (Patología Humana) y del rendimiento de los procedimientos de diagnóstico y tratamiento[20]. La aplicación de Tavlab en estos campos científicos se realiza a través de la resolución de 5 problemas. Los primeros 3 problemas pertenecen al área de la Quimiometría y los 2 restantes al de Medicina Estadística. Concretamente, los problemas 1 y 2 versan sobre el tema de **estadísticas de medidas repetidas** (para revelar la presencia de errores aleatorios en muchos experimentos químicos resulta necesario habitualmente realizar medidas repetidas[27]). El problema 3 trata sobre **contrastes de significación** (para decidir si la diferencia entre la cantidad medida y la cantidad conocida se puede atribuir a errores aleatorios [27]). El problema 4 trata sobre la **correlación y regresión** (determinación del grado de asociación de los datos y la estimación de la mejor línea recta que resume la asociación[28]). Finalmente, el problema 5 versa sobre la **prueba de significación** (aplicación de la prueba t de Student para determinar si una asociación es meramente aparente[28]).

En la Sección 5.1 se revisan los conceptos y métodos fundamentales de la Estadística Descriptiva utilizados en la resolución de los problemas propuestos. La Sección 5.2 contiene la descripción, la solución conceptual y la solución aplicando el entorno Tavlab de estos problemas.

[1] Entorno estadístico basado en *workflows* operado sobre la plataforma Taverna

5.1 Estadística descriptiva

5.1.1 Medidas de tendencia central

Las medidas de tendencia central son tres: la **media**, la **mediana** y la **moda**. Cada una es útil por cuenta propia, pero combinadas ofrecen una imagen muy clara de las características de una muestra.

La **media** es la estadística de uso más común porque es el promedio aritmético de un conjunto de valores. Por tanto, la **media** es el "centro de gravedad" de los datos. Para calcular la **media**, se divide la suma de los elementos en la muestra entre el número de elementos. Por ejemplo, la suma de estos valores: *1 2 3 4 5 6 7 8 9 10*, es 55. Al dividir este valor entre el número de elementos de la muestra, que es 10, arroja la **media**, que es 5.5. Por tanto la fórmula para encontrar la media es:

$$M = \frac{1}{N} \sum_{i=1}^{N} D_i \tag{5.1}$$

donde *Di* representa al *i-ésimo* elemento de la muestra y N es el número de elementos en la muestra.

La **mediana** de una muestra es el valor medio por orden de magnitud. Por ejemplo, en el conjunto de muestras: *1 2 3 4 5 6 7 8 9*, la **mediana** es 5. En el caso de muestras con un número par de elementos, la **mediana** es el promedio de los dos valores centrales. Por ejemplo, para los valores: *1 2 3 4 5 6 7 8 9 10*, la **mediana** es 5.5. En el caso de una muestra que tiene una distribución normal, la **media** y la **mediana** serán similares. Sin embargo, a medida que la distribución de elementos dentro de una muestra se aparta de una curva de distribución normal, la diferencia entre la **mediana** y la **media** aumenta. Una forma fácil de obtener la **mediana** de una muestra es primero ordenar los datos y luego tomar el valor medio.

La **moda** de una muestra es el valor del elemento más frecuente. Por ejemplo, dado la muestra: *1 2 3 3 4 5 6 7 7 7 8 9*, la **moda** es 7 porque ocurre más que cualquier otro elemento. La **moda** no siempre es única. Por ejemplo, dado: *10 20 30 30 40 50 60 60 70*, tanto 30 como 60 ocurren dos veces. Cualquiera podría ser la **moda**. A este tipo de conjunto se la llama bimodal. Un conjunto que sólo tiene una **moda** se le llama unimodal.

5.1.2 Medidas de dispersión

Aunque resulta muy conveniente tener un resumen de "un número", como la **media** y la **mediana**, esto en ocasiones puede confundir, más que informar. Por ejemplo, si una muestra tiene valores agrupados en los extremos, entonces, la **media** y la **mediana** no representan justamente al conjunto. Considere esta muestra: *10 11 9 1 0 2 3 12 11 10*. La **media** es 6.9, pero este valor es apenas

representativo de la muestra porque ningún valor está cerca de él. El problema con la **media** en este caso, es que no comunica información acerca de las variaciones o la **dispersión de los datos**. Para que la **media** tenga más sentido, es necesario saber la cercanía de cada elemento a ella. En esencia, saber la **dispersión de los datos** ayuda a interpretar mejor la **media**, la **mediana** y la **moda**.

Para encontrar la variabilidad de una muestra, debe calcular su **desviación estándar**. La **desviación estándar** se deriva de la **varianza**. Ambos valores representan la **dispersión de los datos**. La **desviación estándar** es el promedio de las distancias entre los valores en la muestra y la media.

La **varianza** se calcula como se muestra a continuación:

$$V = \frac{1}{N-1}\sum_{i=1}^{N}(D_i - M)^2 \tag{5.2}$$

donde N es el número de elementos en el conjunto, M es la **media** y Di es el *i-ésimo* valor de la muestra.

Es necesario elevar al cuadrado la diferencia de la **media** con cada elemeto para producir un resultado positivo. Si no se hiciera así, el resultado siempre sería cero. La **desviación estándar** se deriva al encontrar la raíz cuadrada de la **varianza**. Por tanto, esta es la fórmula de la **desviación estándar**:

$$DE = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(D_i - M)^2} \tag{5.3}$$

5.1.3 Ecuación de regresión

Uno de los usos más comunes de la información estadística es la de predecir eventos futuros. Aunque los datos del pasado no necesariamente predicen eventos futuros, aun así este análisis de tendencias suele resultar muy útil. Tal vez la herramienta más usada para el análisis de tendencias sea la **ecuación de regresión**. Esta es la ecuación de una línea recta que mejor se adapta a los datos y suele conocerse como la línea de regresión, la línea del cuadrado mínimo o la línea más adecuada.

Así, la **ecuación de regresión** tiene la forma de una línea en el espacio bidimensional:

$$Y = a + bX \tag{5.4}$$

donde, X es la variable independiente, Y es la variable dependiente, a es la intercepción en Y y b es la pendiente de la línea. La idea general es encontrar la línea que minimiza la suma de los cuadrados de las desviaciones entre los datos reales y la línea. Por lo tanto, para encontrar la **ecuación de regresión** para un conjunto de valores, primero se debe calcular b con la siguiente fórmula:

$$b = \frac{\sum_{i=1}^{N}(X_i - M_x)(Y_i - M_y)}{\sum_{i=1}^{N}(X_i - M_x)^2} \tag{5.5}$$

donde Mx es la **media** de la coordenada X, y My es la **media** de la coordenada Y. Esta fórmula es equivalente en términos de la **desviación estándar** de X a la siguiente expresión[28]:

$$b = \frac{\sum_{i=1}^{N}(X_i Y_i - N M_x M_y)}{(N-1)DE_x^2} \tag{5.6}$$

donde N es el número de elementos de la muestra y DE_x es la **desviación estándar** de X. Una vez encontrado b, a se calcula con esta fórmula:

$$a = (M_y - bMx) \tag{5.7}$$

5.1.4 Coeficiente de correlación

La forma más común de determinar la correlación de los datos con la línea de regresión consiste en calcular el **coeficiente de correlación** que es un número entre -1 y 1. El **coeficiente de correlación** indica la distancia entre cada punto de datos y la línea. Si el coeficiente es 1, los datos corresponden perfectamente a la línea. Un coeficiente de 0 significa que no existe correlación entre la línea y los datos. (En este caso cualquier línea sería buena.) El signo del **coeficiente de correlación** debe asignarse de acuerdo con el signo de la pendiente de la línea de regresión, que es **b**. Si el coeficiente es positivo, significa que hay una relación directa entre la variable dependiente y la independiente. Si es negativo, entonces existe una relación inversa.

La fórmula para encontrar el **coeficiente de correlación** es:

$$r = \frac{\sum_{i=1}^{N}(X_i - M_x)(Y_i - M_y)}{\sqrt{\sum_{i=1}^{N}(X_i - M_x)^2 \sum_{i=1}^{N}(Y_i - M_y)^2}} \tag{5.8}$$

donde Mx, es la **media** de X y My es la **media** de Y. El signo se asigna de acuerdo con el signo de la pendiente de la línea de regresión. Esta fórmula es equivalente en términos de las **desviaciones estándar** de X y de Y a la siguiente expresión[11]:

$$r = \frac{\sum_{i=1}^{N}(X_i Y_i - N M_x M_y)}{(N-1)DE_x DE_y} \tag{5.9}$$

donde N es el número de elementos de la muestra y, DE_x y DE_y, son respectivamente la **desviación estándar** de X y la **desviación estándar** de Y.

El **coeficiente de correlación** también se utiliza para calcular el **error estándar de la pendiente** de la línea de regresión mediante la siguiente expresión[28]:

$$EE_b = \frac{D_{res}}{\sqrt{\sum_{i=1}^{N}(X_i - M_x)^2}} \tag{5.10}$$

donde D_{res}, la **desviación residual estándar**, se determina con la siguiente fórmula:

$$D_{res} = \sqrt{\frac{DE_y^2(1-r^2)(n-1)}{n-2}} \tag{5.11}$$

5.1.5 Prueba t de Student

Antes de introducir el concepto de la prueba **t de Student** se debe tener claro qué es una hipótesis nula. La hipótesis nula propone algo que inicialmente se presume como verdadero. Se rechaza únicamente cuando se convierte en algo evidentemente falso. Esto es, cuando el investigador tiene un cierto grado de confianza, usualmente del 95 % al 99 %, de que los datos no sustentan la hipótesis nula.

La prueba **t de Student** es una de las técnicas más comunmente usadas para probar una hipótesis nula sobre la base de una diferencia entre las **medias** de dos muestras. La prueba **t de Student** determina la probabilidad de que dos muestras sean iguales con respecto a una variable.

Por ejemplo, supóngase que se colectan datos sobre las alturas de jugadores de basketball y de football y se desea probar la hipótesis nula de que los jugadores no se pueden distingir sólo por su altura. Para probarla (o rechazarla), se comparan las **medias** de las muestras usando la prueba **t de Student**. Una probabilidad de 0.4 indicaría que es 40 % factible que no se pueda distingir un grupo de jugadores de basketball de un grupo de jugadores de football considerando solamente la altura. Si se calcula una probabilidad de 0.05 o menor, entoces se puede rechazar la hipótesis nula (esto es, se puede concluir que los dos grupos de atletas sí pueden ser reconocidos sólo por su altura).

A continuación, se muestra la fórmula para la prueba **t de Student** que se utiliza cuando se cuenta con las **medias** de dos muestras:

$$t = \frac{M_x - M_y}{\sqrt{\frac{V_c}{N_x} + \frac{V_c}{N_y}}} \tag{5.12}$$

donde M_x y M_y son las **medias** de la muestra X y de la muestra Y respectivamente, N_x y N_y son el número de elementos correspondientes a estas muestras y V_c es la **varianza conjunta** que se calcula con la siguiente ecuación[28]:

$$V_c = \frac{(N_x - 1)V_x + (N_y - 1)V_y}{(N_x + N_y) - 2} \tag{5.13}$$

donde V_x y V_y son las varianzas de la muestra X y de la muestra Y respectivamente.

Puesto que el **coeficiente de correlación** nos indica el nivel o grado de asociación entre dos variables, podemos usar la prueba **t de Student** para probar si la asociación es meramente aparente mediante la siguiente fórmula:

$$t = r\sqrt{\frac{N - 2}{1 - r^2}} \tag{5.14}$$

donde r es el **coeficiente de correlación** y N es el número de elementos de la muestra. Esta es la prueba **t de Student** para la correlación.

5.2 Aplicaciones en Quimiometría y Medicina Estadística

A continuación, se resuelven 5 problemas pertenecientes a las áreas de la Quimiometría y la Medicina Estadística. Para cada problema primero se presenta su solución conceptual utilizando las ecuaciones introducidas en la sección anterior y en seguida su solución utilizando el entorno Tavlab en Taverna. En el Capítulo 2 se mostró cómo utilizar Taverna, explicando los pasos que se siguen para construir un *workflow*, suministrarle los datos de entrada, ejecutarlo y guardar los resultados en un archivo. De esta manera, en la resolución de los problemas aplicando el entorno Tavlab se omitirán dichos pasos.

5.2.1 Problema 1: estadísticas de medidas repetidas

Planteamiento y solución conceptual

Un estudiante desea verificar la cantidad de hidróxido sódico que se sabe es de 10.00 ml en una muestra mediante su valoración con ácido clorhídrico. El estudiante repite la valoración 5 veces y obtiene los resultados que se muestran en la *Tabla 5.1.* Determinar la **media** y la **desviación estándar** de estos resultados[27].

Valoraciones	D_i (ml)
1	10.08
2	10.11
3	10.09
4	10.10
5	10.12
Total	50.50

Tabla 5.1: Valoraciones del hidróxido sódico por el estudiante

Solución conceptual:

Para calcular la **media** de los resultados en la *Tabla 5.1*, se utiliza la ecuación (5.1):

$$M = \frac{1}{N}\sum_{i=1}^{N} D_i = \frac{50.50}{5} = 10.1 \ \ ml$$

La **desviación estándar** se obtiene a partir de la ecuación (5.3):

$$DE = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(D_i - M)^2} = \sqrt{\frac{0.001}{4}} = 0.0158 \ \ ml$$

Solución en Taverna

Mediante la ventana localizadora se solicitan los recursos de servicios Web del entorno Tavlab desarrollados para el análisis estadístico de datos experimentales. Ver *Figura 5.1.*

Figura 5.1: Ventana localizadora en la cual se introduce la URL del servicio Web

Los servicios Web solicitados aparecen en el panel de servicios disponibles de Taverna y a continuación se expanden para poder ver sus respectivos métodos. Ver *Figura 5.2*. De esta manera, se pueden seleccionar de este panel los métodos que se necesiten para crear los *workflows* de los problemas propuestos.

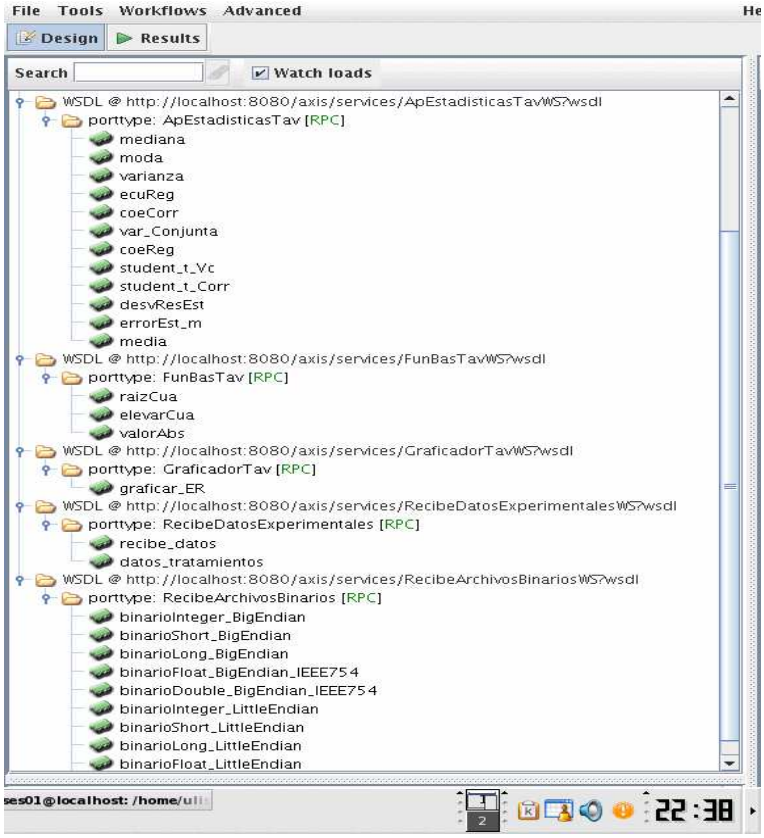


Figura 5.2: Métodos de los servicios Web desarrollados

En seguida, seleccionando los métodos correspondientes del panel de servicios disponibles, se construye el *workflow* que se muestra en la *Figura 5.3* para obtener la media y la desviación estándar de los datos de la *Tabla 5.1*.

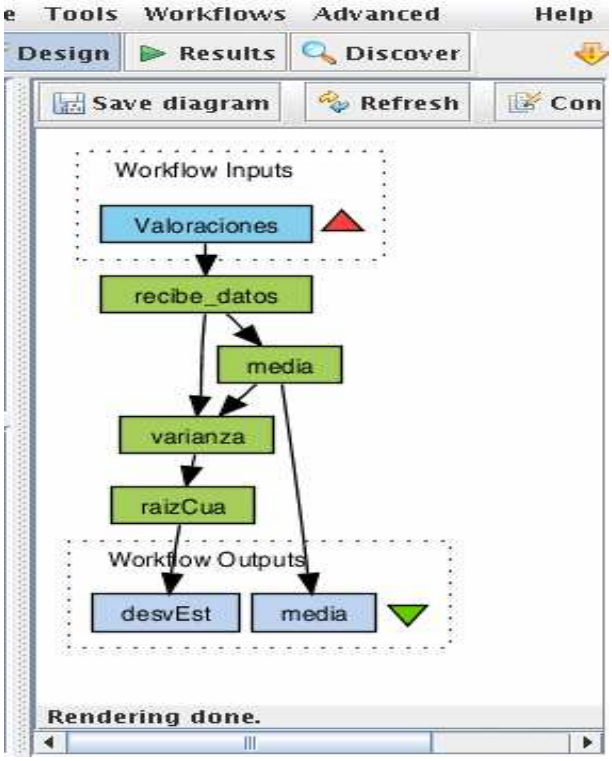


Figura 5.3: *Workflow* del problema 1

Para ilustrar la manera de conectar los métodos, el diagrama de la *Figura 5.4* muestra las conexiones de las entradas y salidas de los íconos representativos de los métodos del *workflow* de la *Figura 5.3*. Para construir un *workflow* el usuario debe referirse al Apendice A conteniendo las descripciones de los íconos representativos de todos los métodos implementados. De esta manera, la *Figura 5.4* muestra el nombre del método correspondiente a cada entrada.

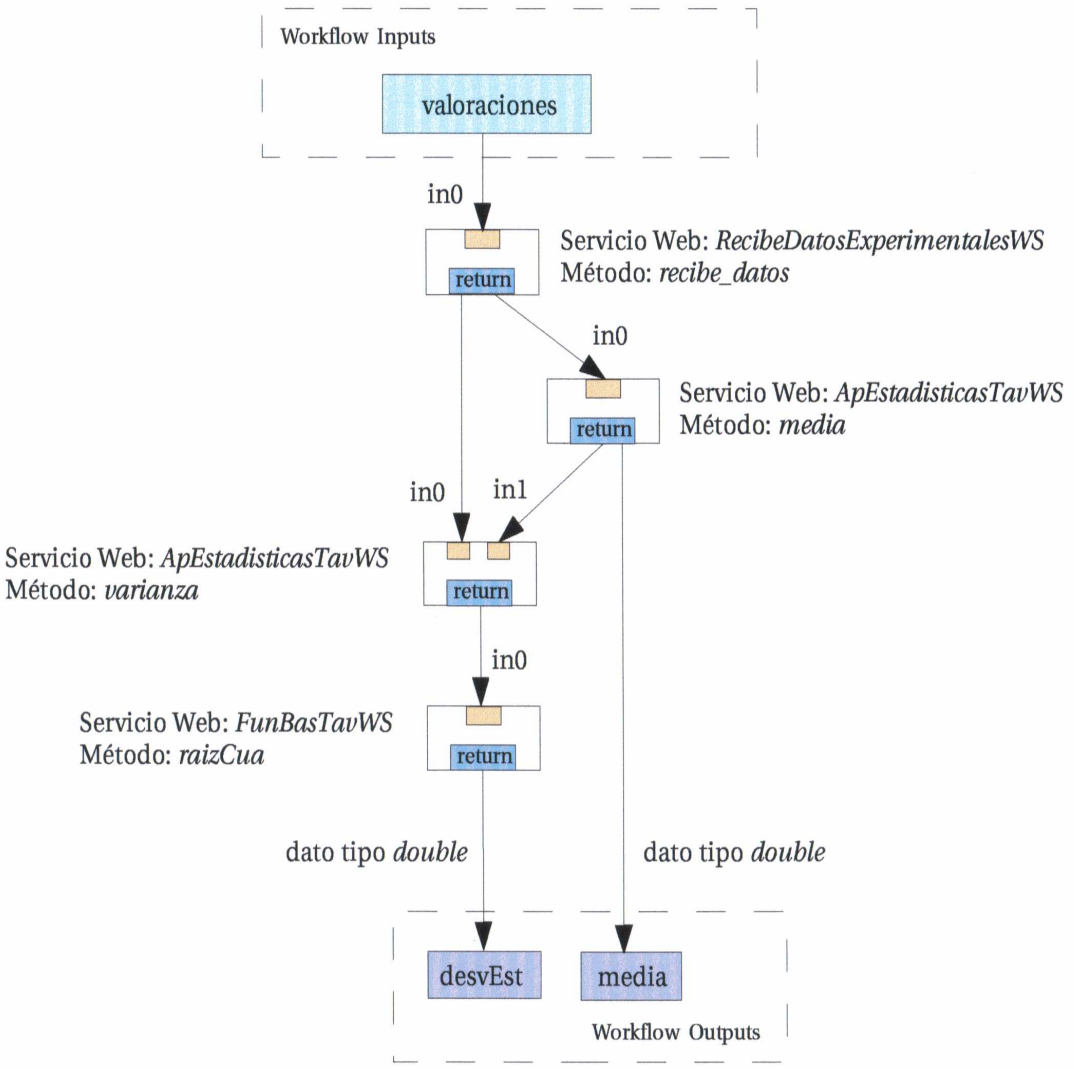


Figura 5.4: Conexiones de las entradas y salidas de los métodos

El *workflow* de la *Figura 5.3* se guarda en un archivo con el nombre Workflow_Prob1.xml para usarlo o compartirlo en aplicaciones futuras. Ver *Figura 5.5.*

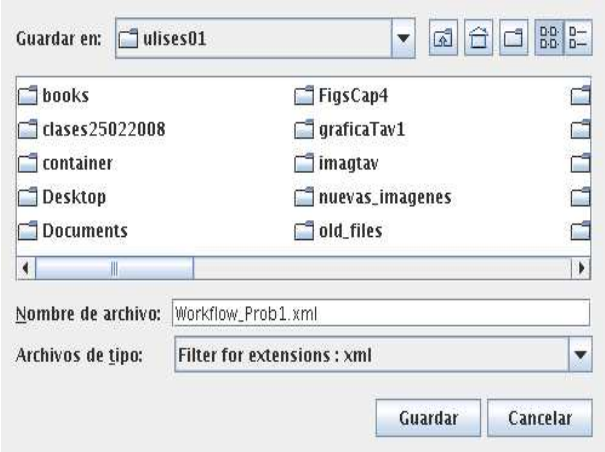


Figura 5.5: Se guarda el *workflow* como Workflow_Prob1.xml

Después de guardar el *workflow*, se alimenta con los datos contenidos en el archivo *valoraciones.txt* que corresponden a los resultados de la *Tabla 5.1.* Ver *Figuras 5.6 y 5.7.*

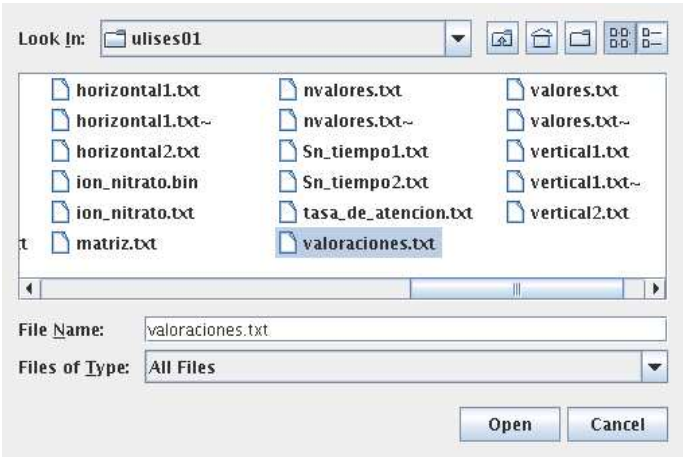


Figura 5.6: Se abre el archivo valoraciones.txt

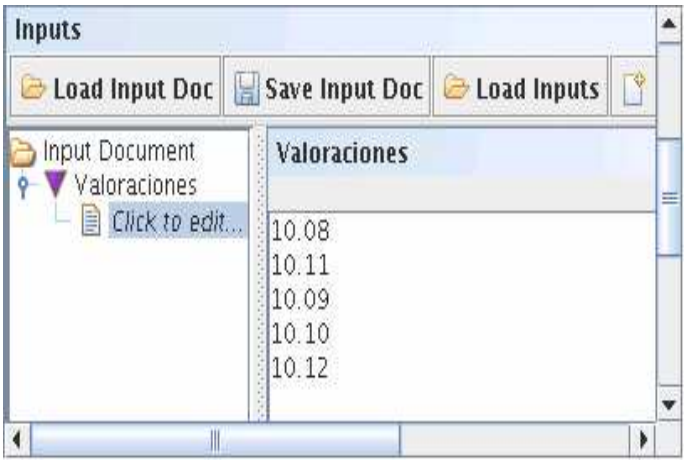


Figura 5.7: Ventana con los datos de entrada para el *workflow*

En seguida, se ejecuta el *workflow* y se obtienen los valores para la **media** y la **desviación estándar** como se muestra en las *Figuras 5.8 y 5.9*.

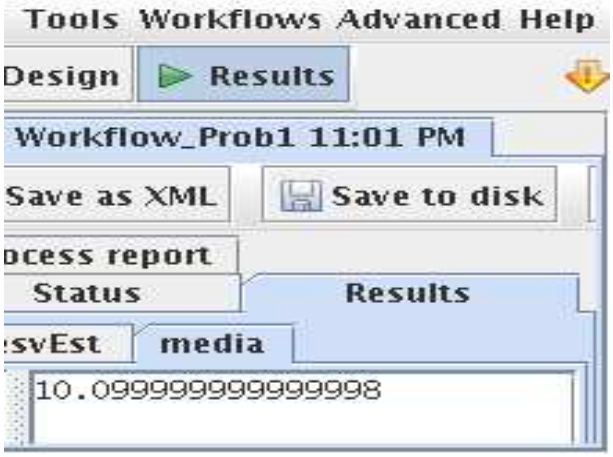


Figura 5.8: Resultado para la media

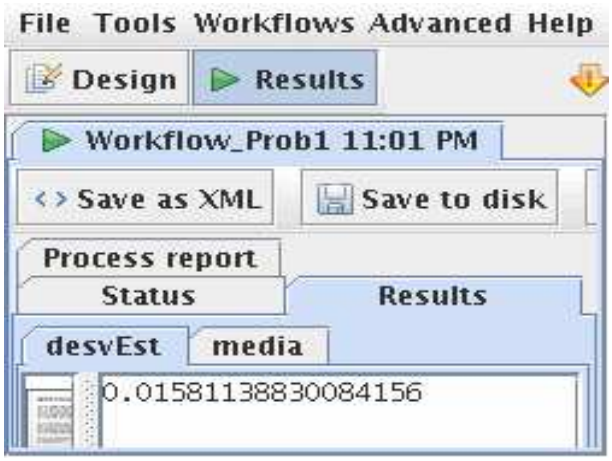


Figura 5.9: Resultado para la desviación estándar

5.2.2 Problema 2: estadísticas de medidas repetidas

Planteamiento y solución conceptual

La *Tabla 5.2* presenta los resultados de 50 determinaciones de la concentración de ion nitrato, con dos cifras significativas, en una muestra concreta de agua[27]. Estos resultados se guardan en un archivo binario llamado *Ion_nitratoBin.bin* en formato Double y sistema Big_Endian para su análisis posterior. Determinar la **media**, la **moda** y la **desviación estándar** de estos resultados.

0.51	0.51	0.51	0.50	0.51	0.49	0.52	0.53	0.50	0.47
0.51	0.52	0.53	0.48	0.49	0.50	0.52	0.49	0.49	0.50
0.49	0.48	0.46	0.49	0.49	0.48	0.49	0.49	0.51	0.47
0.51	0.51	0.51	0.48	0.50	0.47	0.50	0.51	0.49	0.48
0.51	0.50	0.50	0.53	0.52	0.52	0.50	0.50	0.51	0.51

Tabla 5.2: Resultados de 50 determinaciones de concentración de ion nitrato en $\mu g\ ml^{-1}$

Solución conceptual:

La **media** de los resultados en la *Tabla 5.2* se calcula con la ecuación (5.1):

$$M = \frac{1}{N}\sum_{i=1}^{N} D_i = \frac{24.99}{50} = 0.4998 \ \mu g \ ml^{-1}$$

El valor que se repite con más frecuencia en la *Tabla 5.2* es el 0.51, así:

$$moda = 0.51 \ \mu g \ ml^{-1}$$

Utilizando ahora la ecuación (5.3) se obtiene la **desviación estándar**:

$$DE = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(D_i - M)^2} = \sqrt{\frac{0.013298}{49}} = 0.01647 \ \mu g \ ml^{-1}$$

Solución en Taverna

Como los resultados de la *Tabla 5.2* se han guardado en el archivo binario *Ion_nitratoBin.bin* en formato Double y sistema Big_Endian se utiliza el método *binarioDouble_BigEndian_IEE754* del Servicio Web *RecibeArchivosBinariosWS* para transformar los datos en formato ascii y entregar a la entrada del *workflow* de la *Figura 5.10* el correspondiente arreglo tipo string de datos. El *workflow* de la *Figura 5.10* representa el "programa gráfico" solución del problema 2. Como se aprecia en la *Figura 5.10*, el ícono del método *binarioDouble_BigEndian_IEEE754* se encuentra precisamente entre el ícono de la entrada representando al archivo *Ion_nitratoBin* y los íconos de los métodos estadísticos que procesan los datos. Como se mencionó en el Capítulo 3, las entradas válidas para los métodos implementados en Tavlab (corresondientes a los niveles II, III y IV) son arreglos tipo string de datos o datos individuales de cualquier tipo (integer, double, float, etc).

A continuación, se guarda el *workflow* en el archivo Workflow_Prob2.xml tal como se hizo en el problema anterior y se alimenta con los datos del archivo binario *Ion_nitratoBin.bin*. Como se muestra en la *Figura 5.11*, los datos binarios de entrada no se pueden visualizar, por lo que aparece la leyenda *No renderer*.

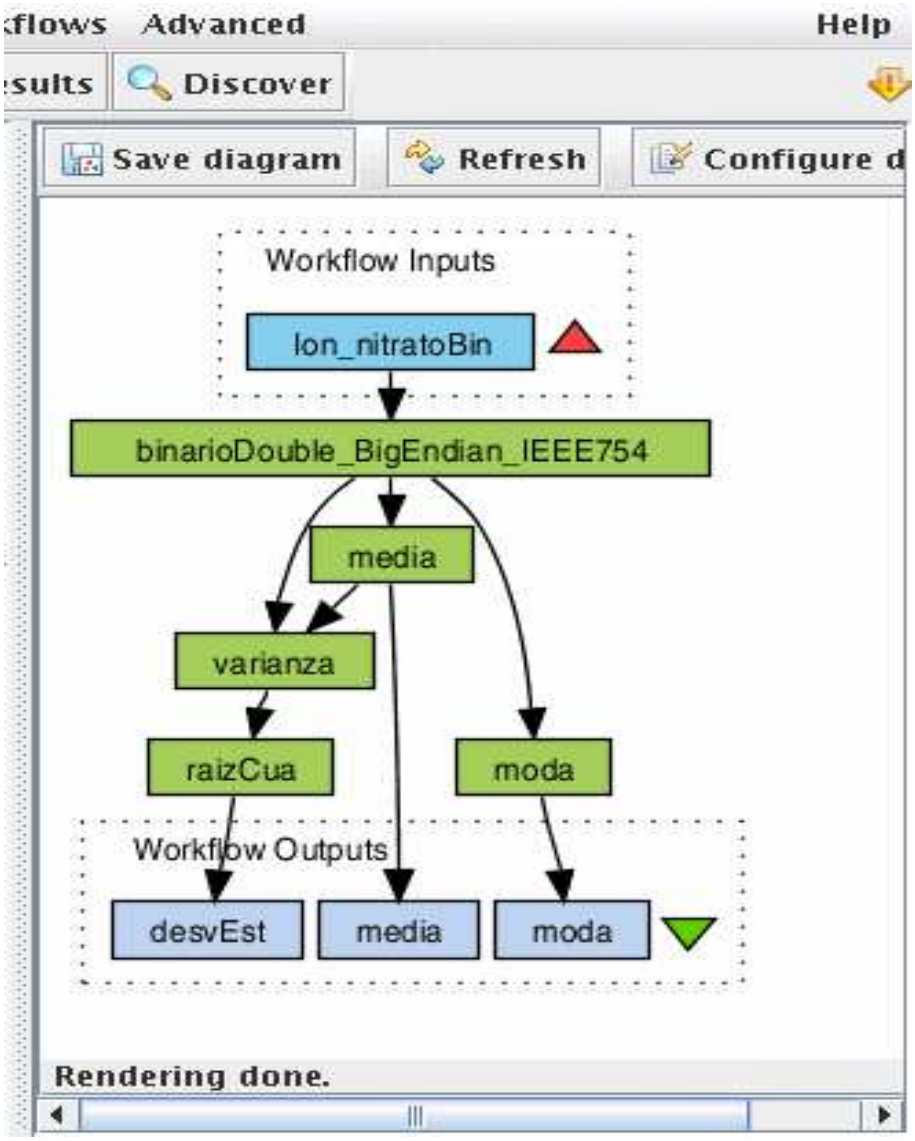


Figura 5.10: *Workflow* para el problema 2

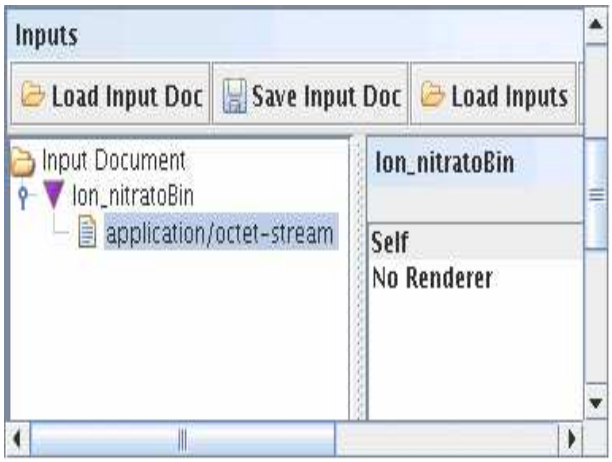


Figura 5.11: Ventana para los datos de entrada del *workflow*

Se ejecuta el *workflow* y se obtienen los resultados que se muestran en las *Figuras 5.12, 5.13 y 5.14*.

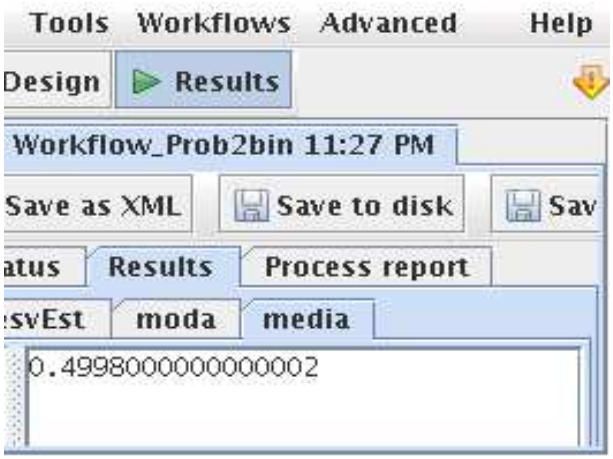


Figura 5.12: Resultado para la media

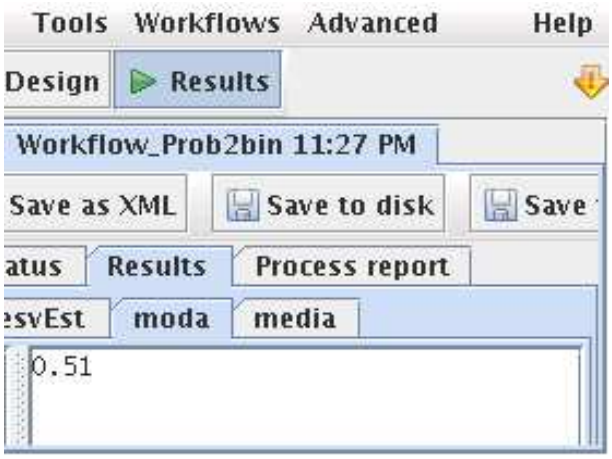


Figura 5.13: Resultado para la moda

Figura 5.14: Resultado para la desviación estándar

5.2.3 Problema 3: contrastes de significación

Planteamiento y solución conceptual

En una serie de experimentos para la determinación de estaño en productos alimenticios, las muestras fueron llevadas a ebullición con ácido clorhídrico (HCL) a reflujo para diferentes tiempos. Los resultados se presentan en la *Tabla 5.3*[27]. Aplicar la **prueba t de Student** a la diferencia de las **medias** muestrales.

Tiempo de reflujo (min)	Estaño encontrado ($mg\,Kg^{-1}$)
30	55, 57, 59, 56, 56, 59
75	57, 55, 58, 59, 59, 59

Tabla 5.3: Resultados de la determinación de estaño

Solución conceptual:

Primero, se obtienen las **medias** para los dos tiempos mediante la ecuación (5.1):

$$M_x = \frac{1}{N}\sum_{i=1}^{N} D_{ix} = \frac{342}{6} = 57.00\ \ mg\,Kg^{-1}$$

$$M_y = \frac{1}{N}\sum_{i=1}^{N} D_{iy} = \frac{347}{6} = 57.83\ \ mg\,Kg^{-1}$$

Luego, se encuentra la **varianza** respectiva a cada tiempo con la ecuación (5.2):

$$V_x = \frac{1}{N-1}\sum_{i=1}^{N}(D_i - M_x)^2 = \frac{14.00}{5} = 2.80\ \ mg\,Kg^{-1}$$

$$V_y = \frac{1}{N-1}\sum_{i=1}^{N}(D_i - M_y)^2 = \frac{12.83}{5} = 2.57\ \ mg\,Kg^{-1}$$

A continuación, se calcula **el valor conjunto para la varianza** mediante la ecuación (5.11):

$$V_c = \frac{(N_x - 1)V_x + (N_y - 1)V_y}{(N_x + N_y) - 2} = \frac{(5 \times 2.80 + 5 \times 2.57)}{10} = 2.685\ \ mg\,Kg^{-1}$$

Finalmente, se aplica la ecuación (5.10) y se obtiene el valor **t de Student**:

$$t = \frac{M_x - M_y}{\sqrt{\frac{V_c}{N_x} + \frac{V_c}{N_y}}} = \frac{57.00 - 57.83}{1.64\sqrt{\frac{1}{6} + \frac{1}{6}}} = -0.88$$

Solución en Taverna

Para este problema se crea el *workflow* que se muestra en la *Figura 5.15* y se guarda con el nombre de archivo Workflow_Prob3.xml.

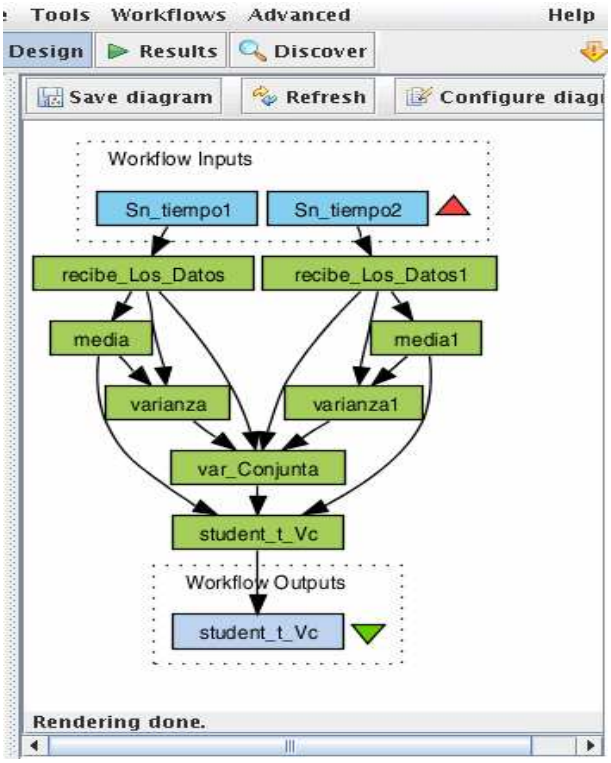


Figura 5.15: *Workflow* para la prueba t de Student

A continuación, se le suministran los datos de entrada que se encuentran en los archivos *Sn_tiempo1* y *Sn_tiempo2*. Estos archivos corresponden a los resultados de la determinación de estaño para cada uno de los tiempos que se muestran en la *Tabla 5.3*. Ver *Figura 5.16*.

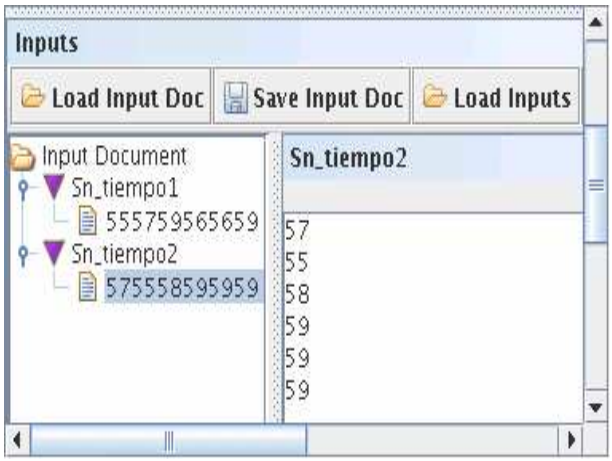


Figura 5.16: Se suministran los datos de entrada para el *workflow*

Después de alimentar el *workflow*, se ejecuta y se obtiene el valor de la **prueba t de Student** como se muestra en la *Figura 5.17*.

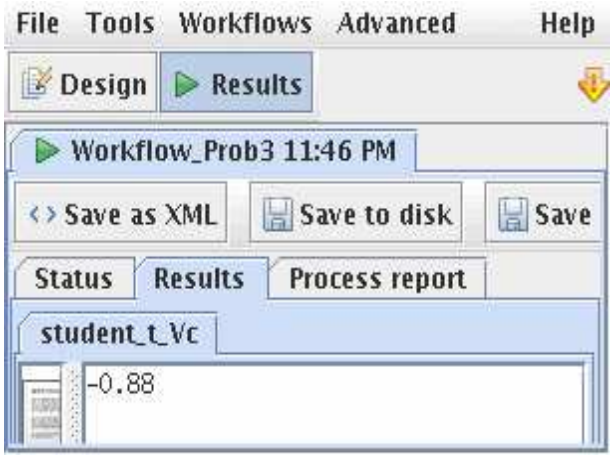


Figura 5.17: Resultado de la prueba t de Student

5.2.4 Problema 4: correlación y regresión

Planteamiento y solución conceptual

Se efectuó un estudio relativo a la tasa de atención de un hospital que presta servicio a comunidades localizadas en 16 áreas geográficas diferentes, en un periodo de tiempo fijo. La distancia desde el hospital al centro de cada área geográfica está medida en kilómetros. La *Tabla 5.4* presenta los resultados. Calcular el **coeficiente de correlación** y encontrar la **ecuación de regresión**[28].

Área geográfica	Distancia (Km)	Tasa de atención (%)
1	6.8	21
2	10.3	12
3	1.7	30
4	14.2	8
5	8.8	10
6	5.8	26
7	2.1	42
8	3.3	31
9	4.3	21
10	9.0	15
11	3.2	19
12	12.7	6
13	8.2	18
14	7.0	12
15	5.1	23
16	4.1	34

Tabla 5.4: Distancia y tasa de atención en 16 áreas geográficas

Solución conceptual:

Se calculan las **medias** para la distancia y la tasa de atención empleando la ecuación (5.1):

$$M_x = \frac{1}{N}\sum_{i=1}^{N} D_{ix} = \frac{106.6}{16} = 6.6625 \ \ Km$$

$$M_y = \frac{1}{N}\sum_{i=1}^{N} D_{iy} = \frac{328}{16} = 20.5 \ \%$$

Se obtienen ahora sus respectivas **desviaciones estándar** mediante la ecuación (5.3):

$$DE_x = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(D_i - M_x)^2} = \sqrt{\frac{203.12}{15}} = 3.67967 \ \ Km$$

$$DE_y = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(D_i - M_y)^2} = \sqrt{\frac{1542}{15}} = 10.139 \ \%$$

Utilizando la ecuación (5.9) se calcula el **coeficiente de correlación** como sigue:

$$r = \frac{\sum_{i=1}^{N} X_i Y_i - N M_x M_y}{(N-1) DS_x DS_y} = \frac{1713.3 - 2185.3}{15(3.67967)(10.139)} = -0.843$$

Combinando las ecuaciones (5.6) y (5.9) se obtiene la siguiente expresión para la pendiente de la **línea de regresión**:

$$b = r\frac{DE_y}{DE_x} = -0.843\frac{10.139}{3.67967} = -2.32$$

Una vez encontrado b, la ordenada en el origen a de la **línea de regresión** se calcula con la ecuación (5.7):

$$a = (M_y - bM_x) = 20.5 - (-2.32)(6.6625) = 35.957 \approx 36$$

Finalmente, al sustituir a y b en (5.4) se encuentra la **ecuación de regresión**:

$$y = 36 - 2.32x$$

Solución en Taverna

La *Figura 5.18* muestra el *workflow* que se construyó para la solución de este problema.

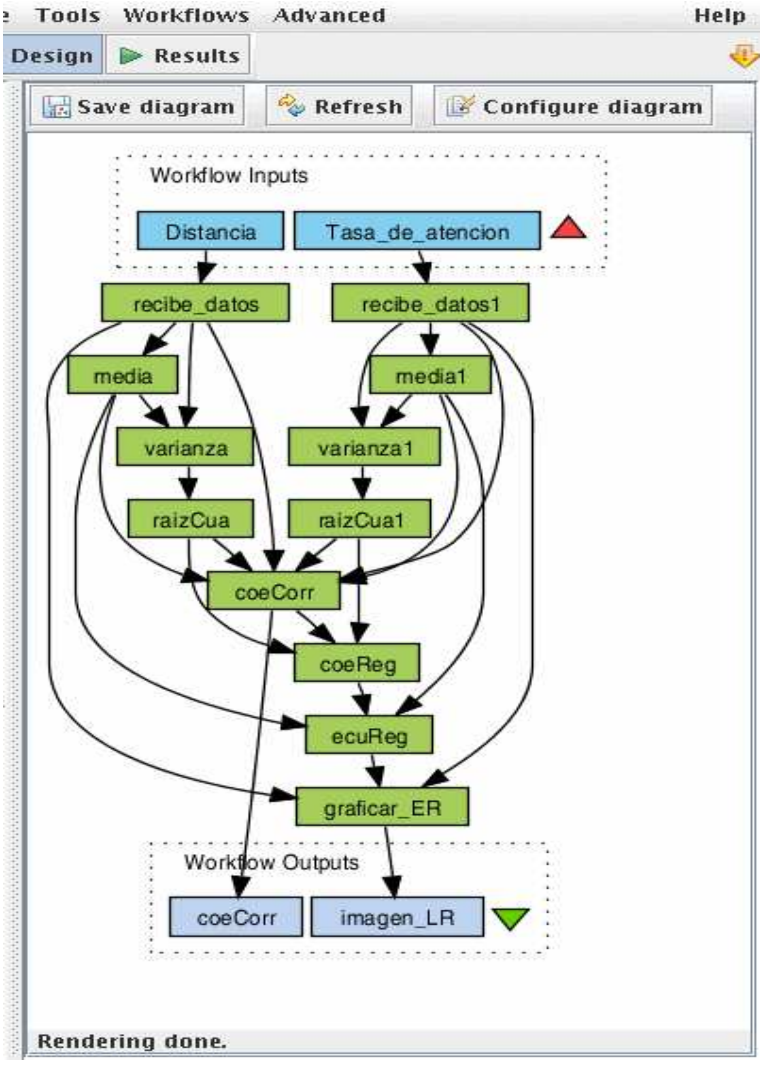


Figura 5.18: *Workflow* para el problema 4

A continuación, la salida **imagen_LR** correspondiente al método **graficar_ER** debe configurarse para recibir el tipo de datos MIME **image/jpeg**. Esto se hace de la siguiente manera: en la ventana del **explorador** seleccionar con el botón derecho del mouse la salida **imagen_LR** y en el menú que se presenta elegir **Metadata for output imagen_LR**. En seguida, aparecerá una ventana en la cual se debe introducir el tipo de dato MIME **image/jpeg** como se muestra en la *Figura 5.19*. Después, oprimir **ENTER** para que este tipo de dato quede registrado.

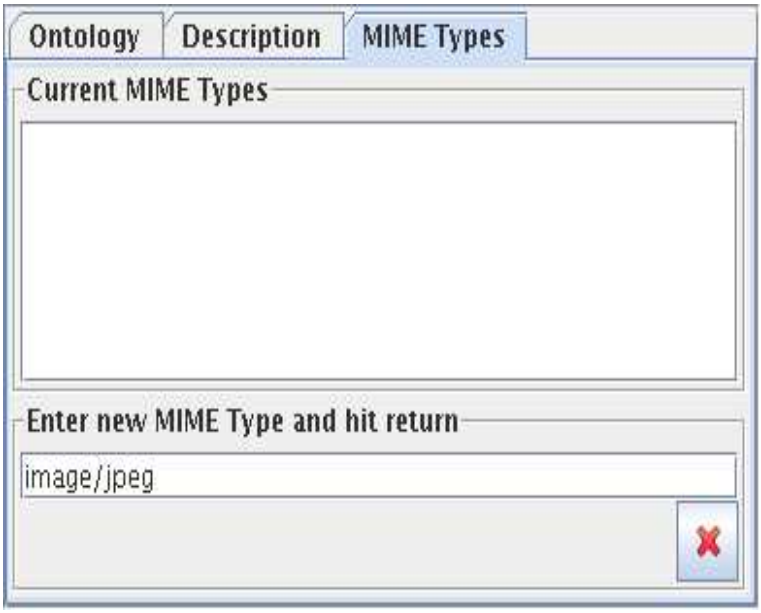


Figura 5.19: Ventana donde se introducen los tipos de datos MIME

Luego, el *workflow* se guarda con el nombre de archivo Workflow_Prob4.xml y se alimenta con los datos de entrada contenidos en los archivos *distancias.txt* y *tasa de atención.txt* . Ver *Figura 5.20*.

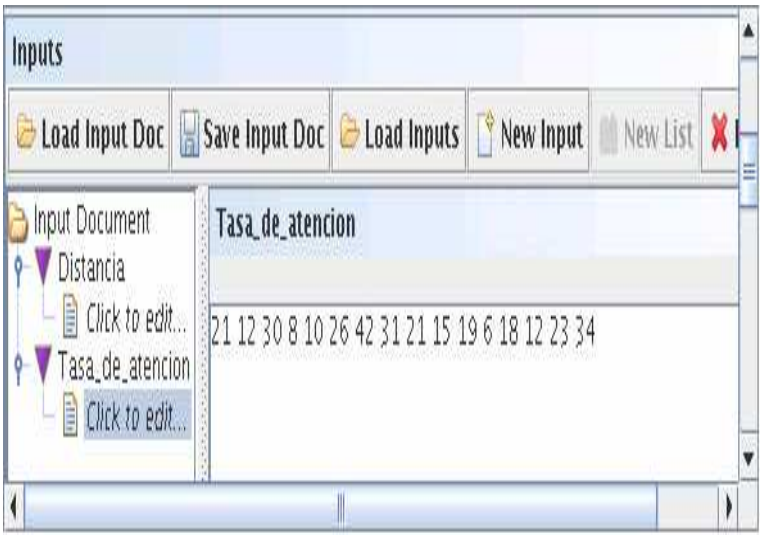


Figura 5.20: Datos de entrada para el *workflow*

En seguida, se ejecuta el *workflow* y se obtienen los resultados que se presentan en las *Figuras 5.21 y 5.22*.

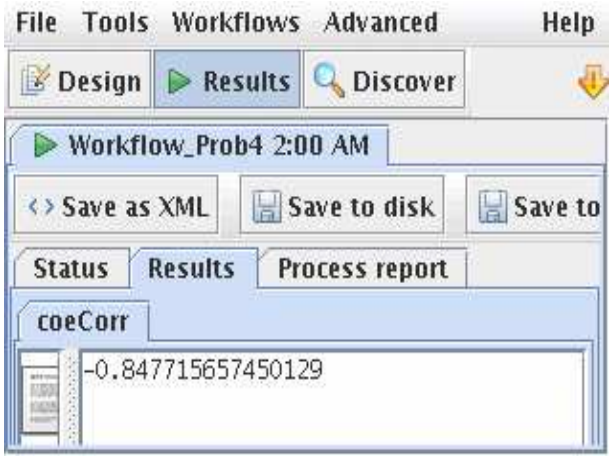


Figura 5.21: Se obtiene el coeficiente de correlación

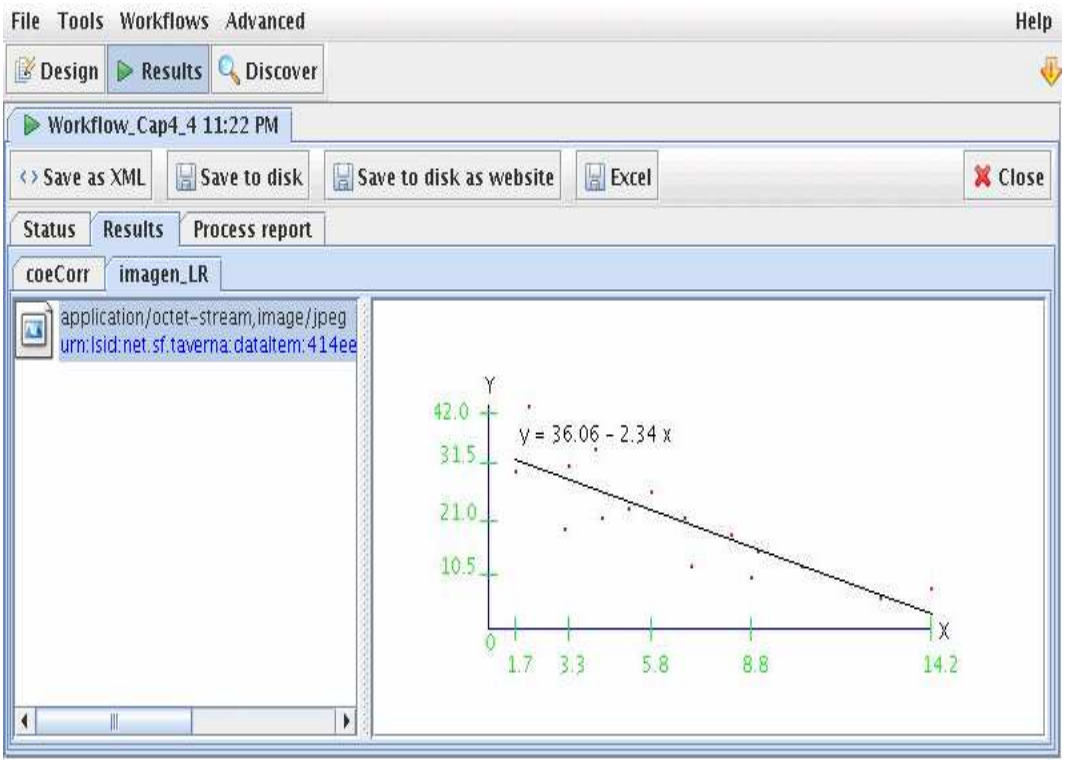


Figura 5.22: Gráfica de la línea de regresión sobre el diagrama de dispersión

5.2.5 Problema 5: prueba de significación

Planteamiento y solución conceptual

Un investigador pediatra ha registrado las medidas del espacio anatómico pulmonar (en ml) y las alturas (en cm) de 15 niños. Estos datos se muestran en la *tabla 4.6*. Se desea conocer la **correlación** que existe entre estos dos parámetros y aplicar la **prueba t de Student** para la correlación. Así también, desea obtener tanto el **coeficiente** como la **línea de regresión** y calcular el **error estándar** de la pendiente[28].

Niño	Altura (cm)	EAP (ml)
1	110	44
2	116	31
3	124	43
4	129	45
5	131	56
6	138	79
7	142	57
8	150	56
9	153	58
10	155	92
11	156	78
12	159	64
13	164	88
14	168	112
15	174	101

Tabla 5.5: Alturas y espacio anatómico pulmonar en 15 niños

Solución conceptual:

Se calculan las **medias** para la altura y el espacio anatómico pulmonar empleando la ecuación (5.1):

$$M_x = \frac{1}{N}\sum_{i=1}^{N} D_{ix} = \frac{2169}{15} = 144.6 \ cm$$

$$M_y = \frac{1}{N}\sum_{i=1}^{N} D_{iy} = \frac{1004}{15} = 66.933 \ ml$$

Se encuentran ahora sus respectivas **desviaciones estándar** mediante la ecuación (5.3):

$$DE_x = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(D_i - M_x)^2} = \sqrt{\frac{5251.6177}{14}} = 19.3679 \ cm$$

$$DE_y = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(D_i - M_y)^2} = \sqrt{\frac{7828.9258}{14}} = 23.6476 \ ml$$

El **coeficiente de correlación** se obtiene utilizando los resultados anteriores en la ecuación (5.9) como sigue:

$$r = \frac{\sum_{i=1}^{N} X_i Y_i - N M_x M_y}{(N-1) DS_x DS_y} = \frac{5426.6}{14(19.3679)(23.6476)} = 0.846$$

A continuación, se utiliza la ecuación (5.12) que es **prueba t de Student** para la correlación:

$$t = r\sqrt{\frac{N-2}{1-r^2}} = 0.846\sqrt{\frac{15-2}{1-0.846^2}} = 5.72$$

Combinando las ecuaciones (5.6) y (5.9) se obtiene la siguiente expresión para la pendiente de la **línea de regresión**:

$$b = r\frac{DE_y}{DE_x} = 0.846\frac{23.6476}{19.3679} = 1.033$$

A b también se le llama **coeficiente de regresión**. Una vez encontrado b, la ordenada en el origen a de la **línea de regresión** se calcula con la ecuación (5.7):

$$a = (M_y - bM_x) = 66.933 - (1.033)(144.6) = -82.4$$

Luego, al sustituir a y b en (5.4) se encuentra la **línea de regresión**:

$$y = -82.4 + 1.033x$$

Para encontrar el **error estándar** de la pendiente, se debe calcular primero la **desviación residual estándar** mediante la ecuación (5.14):

$$D_{res} = \sqrt{\frac{DE_y^2(1-r^2)(n-1)}{n-2}} = \sqrt{\frac{23.6476^2(1-0.846^2)(15-1)}{15-2}} = 13.08445$$

Finalmente, se sustituye el valor anterior en la ecuación (5.13) para obtener el **error estándar** de la pendiente:

$$EE_b = \frac{D_{res}}{\sqrt{\sum_{i=1}^{N}(D_{ix} - M_x)^2}} = \frac{13.08445}{72.4680} = 0.18055$$

Solución en Taverna

El *workflow* que se creó para este problema se guardó con el nombre de archivo Workflow_Prob5.xml. De manera que, para colocar el *workflow* en la ventana de diagramas y ejecutarlo, abrimos el archivo de la siguiente manera: se selecciona del menú principal la opción **File** y luego **Open workflow from a file**. A continuación, se presentará una ventana selectora en la cual se introduce el nombre de archivo Workflow_Prob5.xml como se muestra en la *Figura 5.23*. Después, se presiona el botón **Open** y el *workflow* aparecerá en la ventana de diagramas. Ver *Figura 5.24*. De manera semejante a como se hizo en el problema anterior, la salida **imagen_LR** correspondiente al método **graficar_ER** debe configurarse para recibir el tipo de datos MIME **image/jpeg** antes de ejecutar el *workflow*.

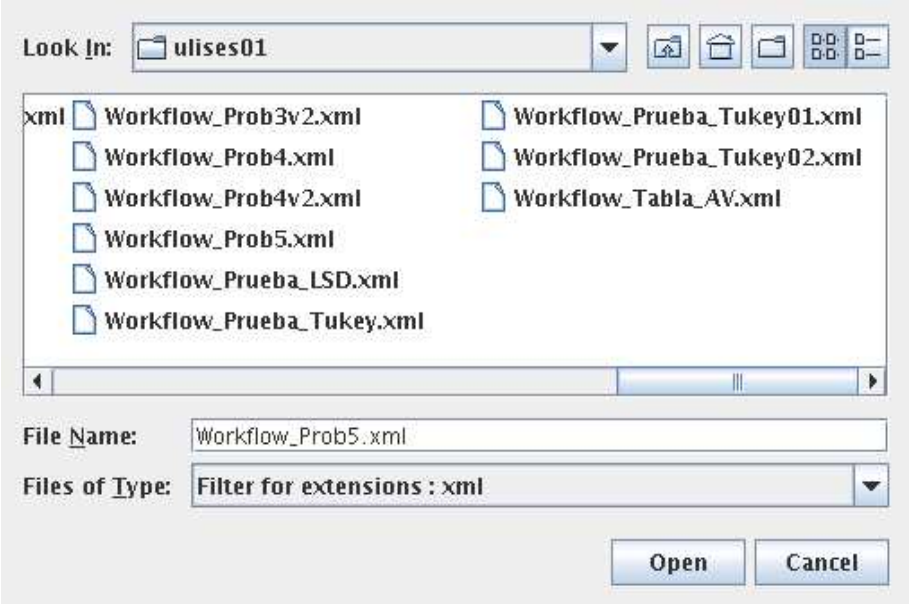


Figura 5.23: Se abre el *workflow* del archivo Workflow_Prob5.xml

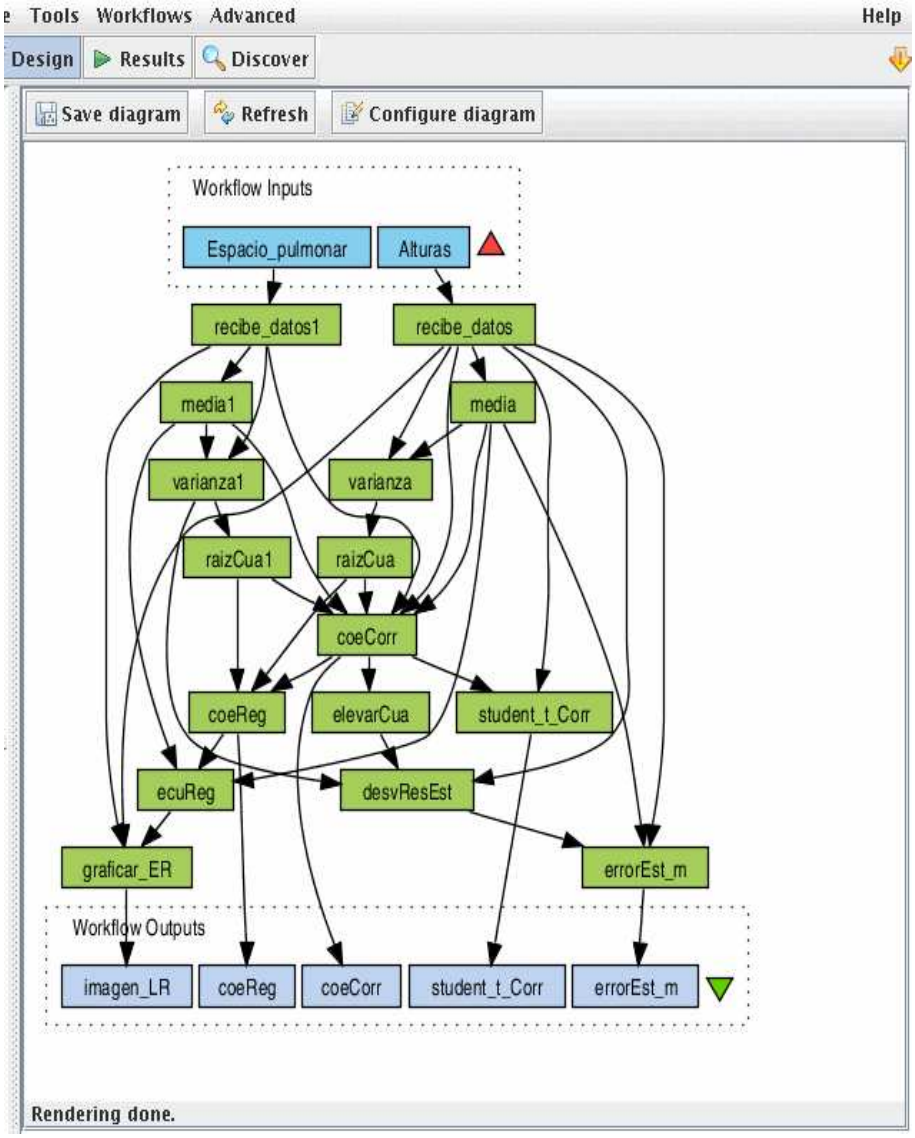


Figura 5.24: *Workflow* para el problema 5

En seguida, se le suministran al *workflow* los datos de entrada contenidos en los archivos *Alturas.txt* y *Espacio_pulmonar.txt*. Ver *Figura 5.25*.

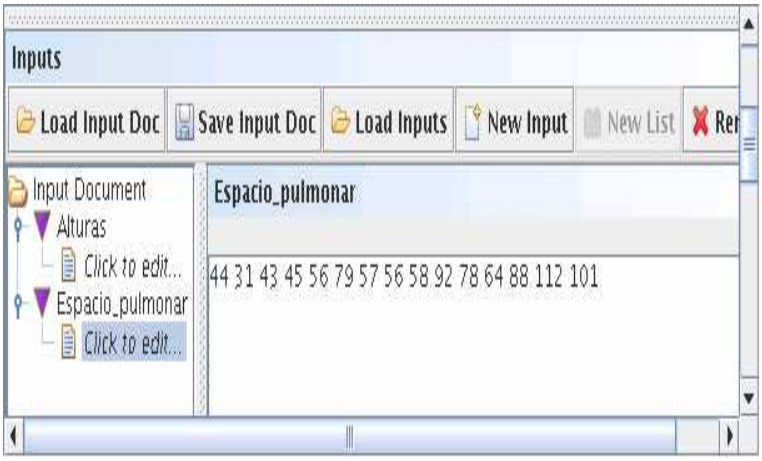


Figura 5.25: Se suministran los datos de entrada para el *workflow*

Finalmente, se ejecuta el *workflow* y se obtienen los resultados que se presentan en las *Figuras* de la *5.26* a la *5.30*.

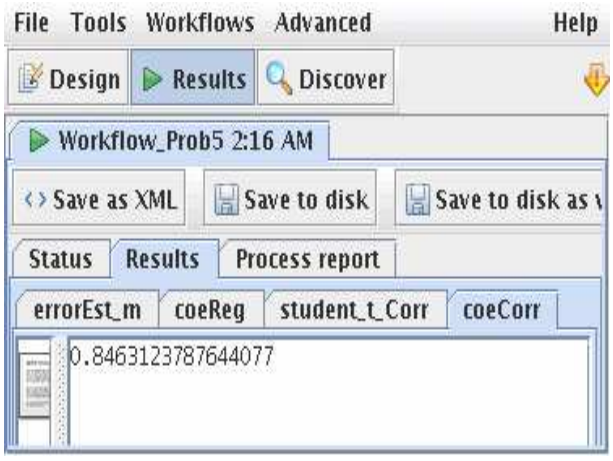


Figura 5.26: Resultado para el coeficiente de correlación

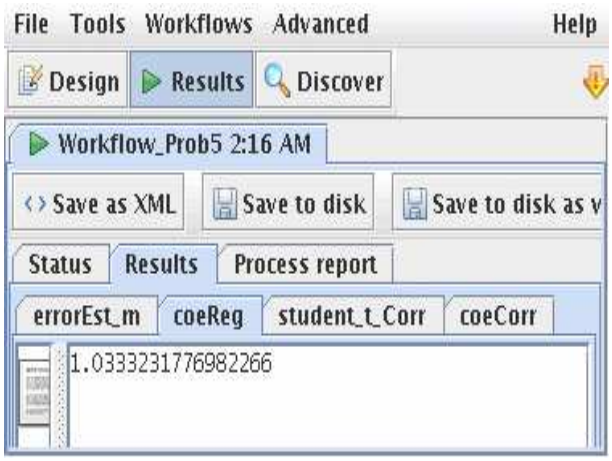


Figura 5.27: Resultado para el coeficiente de regresión

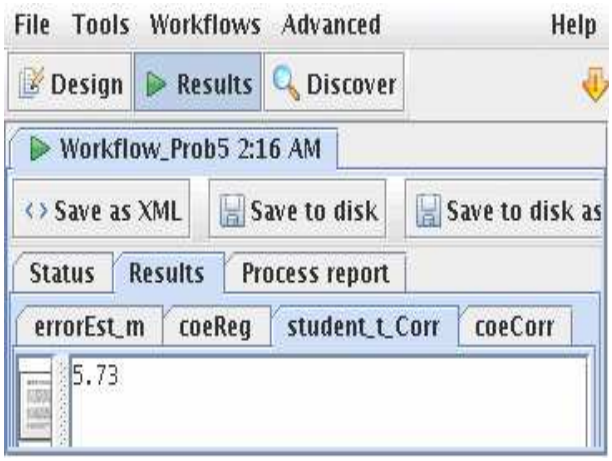


Figura 5.28: Resultado de la prueba t de Student para la correlación

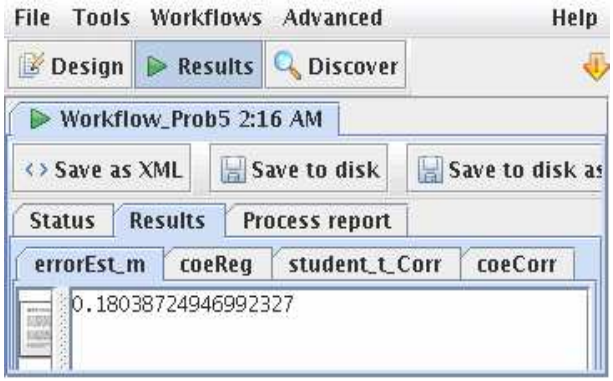


Figura 5.29: Resultado para el error estándar de la pendiente

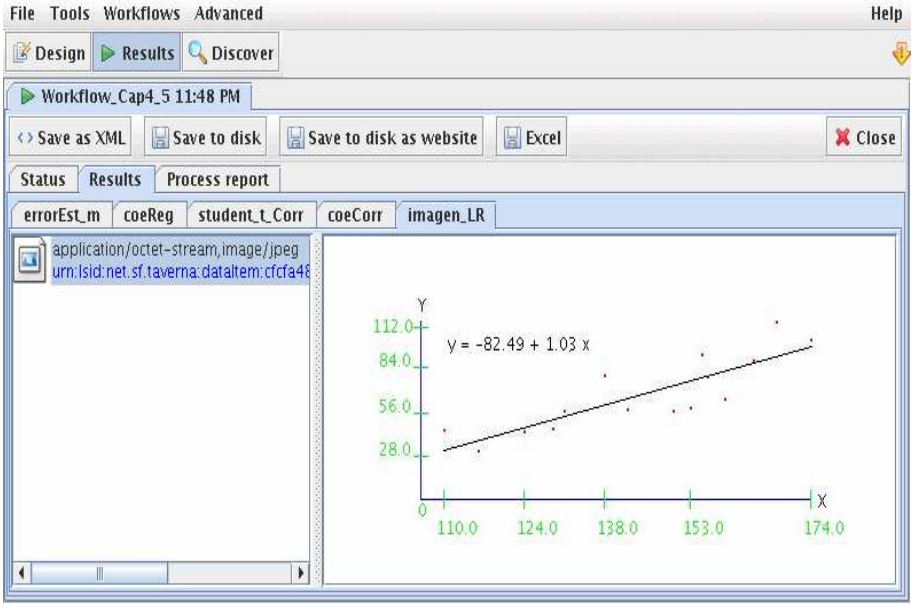


Figura 5.30: Gráfica de la línea de regresión sobre el diagrama de dispersión

5.3 Sumario

En este capítulo se explicó cómo aplicar los recursos de servicios Web del entorno Tavlab a la resolución de problemas de análisis estadístico de datos experimentales. Los problemas propuestos pertenecen a los campos de la Quimiometría y la Medicina Estadística, ambas, ciencias experimentales que se apoyan en gran medida en la Estadística para llevar a cabo el análisis de resultados de sus experimentos. Como preámbulo, se revisaron al inicio del capítulo los conceptos y las ecuaciones principales de la Estadística Descriptiva utilizados para la resolución de los problemas propuestos.

Así también, el grado de dificultad de los problemas propuestos aumenta gradualmente con la finalidad de hacer más intuitiva la utilización del entorno estadístico Tavlab desarrollado en este trabajo de investigación. Los *workflows* creados en este capítulo representan "programas gráficos" solución de los problemas propuestos.

Capítulo 6

Conclusiones y trabajo futuro

La principal contribución de nuestro trabajo de investigación consiste en la implementación de un entorno de servicios Web con componentes de software (métodos) diseñados para realizar aplicaciones de análisis estadístico y de análisis de varianza de datos experimentales, los cuales pueden ser integrados, reutilizados y compartidos para construir *workflows* en la plataforma Taverna. Hemos llamado a este entorno estadístico basado en *workflows* con Taverna, Tavlab (Capítulo 3).

Taverna es un ambiente gráfico de resolución de problemas en forma distribuida bien establecida y ampliamente utilizada para llevar a cabo experimentos con *workflows* en la *e-Sicience*. En Taverna, todas las etapas computacionales de los *workflows* son componentes de software ofrecidos por servicios Web (Capítulo 2).

Los servicios Web de Tavlab están organizados en cuatro niveles de aplicación: el nivel destinado al manejo de datos de entrada, el nivel de aplicaciones estadísticas, el nivel de apoyo y consulta, y el nivel de presentación de resultados (Capítulo 3). Tavlab es un entorno abierto, de manera que pueden adicionarse otros servicios Web conforme se requieran en el futuro.

El nivel destinado al manejo de datos de entrada consiste de 2 servicios Web con 12 métodos para leer archivos de datos sin importar su disposición y formato (ascii o binario), y hacerlos disponibles de manera adecuada a las entradas de los *workflows* de análisis estadístico.

El nivel de aplicaciones consiste de un servicio Web con 12 métodos mediante los cuales es posible realizar diferentes tipos de análisis propios de la Estadística Descriptiva como son: obtener las medidas de tendencia central y de dispersión, la ecuación de regresión, el coeficiente de correlación, aplicar la prueba t de Student, etc. Así también, consiste de un servicio Web con 6 métodos para realizar el análisis de varianza de datos experimentales y aplicar las pruebas de Tukey y de Fisher a pares de medias.

El nivel de apoyo y consulta cuenta con un servicio Web con 3 métodos para realizar cálculos aritméticos y un servicio Web con 2 métodos para realizar consultas a tablas de estadísticos necesarias en el análisis de varianza.

El último nivel, presentación de resultados, consiste de 2 servicios Web con 3 métodos destinados a graficar la línea de regresión y la dispersión de los datos de

entrada, así como las tablas de resultados del análisis de varianza.

Para implementar Tavlab se requirió conocer y aplicar los estándares abiertos: lenguaje de marcado extensible XML (*eXtensive Markup Language*), protocolo simple de acceso a objetos SOAP (*Simple Object Access Protocol*), descriptor de activación de servicios Web WSDD (*Web Services Deployment Descriptor*) y el lenguaje de descripción de servicios Web WSDL (*Web Services Description Language*). Estos estándares permiten a los servicios Web, respectivamente, estructurar información, intercambiar mensajes, activar componentes de software (métodos) y describir aplicaciones (Capítulo 3). También fue necesario instalar un servidor Apache-Axis para utilizarlo como contenedor de servicios Web.

Para realizar los programas de las aplicaciones estadísticas ofrecidas por los servicios Web fue indispensable conocer con profundidad los parádigmas de la programación orientada a objetos, en particular del lenguaje Java. Un aspecto importante de nuestra investigación se centró, precisamente, en la programación en lenguaje Java de aplicaciones que generaran gráficas para mostrarlas en Taverna (Capítulo 2).

Nuestro entorno Tavlab aporta varios beneficios como son, construir *workflows* a partir de servicios Web distribuidos en varios sitios en Internet, así, los datos experimentales pueden localizarse no sólo en el sitio de trabajo, sino también en cualquier otro, aprovechando una mayor capacidad de cómputo (Capítulo 4).

Para validar nuestra implementación, se resolvieron problemas pertenecientes a los campos de la Biotecnología, la Quimiometría y la Medicina Estadística. La Biotecnología es el campo del conocimiento que integra el uso de las ciencias de la Bioquímica y la Microbiología con la ingeniería para llevar a cabo aplicaciones controladas y deliberadas de las capacidades de agentes biológicos, (microorganismos, cultivos celulares, etc) para uso industrial. La Quimiometría es la disciplina química que se enfoca en la aplicación de métodos matemáticos o estadísticos sobre datos químicos. La Medicina Estadística se apoya en métodos estadísticos para la investigación de los mecanismos de la enfermedad (Patología Humana) y del rendimiento de los procedimientos de diagnóstico y tratamiento. Los *workflows* creados son "programas gráficos" que resuelven los problemas propuestos (Capítulos 4 y 5).

Los ambientes distribuidos de resolución de problemas DPSE (*Distributed Problem Solving Environment*) son necesarios para que recursos computacionales distribuidos/paralelos se pongan al alcance de la mano de científicos de diferentes áreas sin que tengan que conocer mucho de computación (Sistemas Operativos, Redes, Servidores, Web, Java, servicios Web, etc), y así puedan concentrarse en la resolución de los problemas de su área. Taverna es un DPSE que fomenta la colaboración y compartición de recursos entre diferentes personas. La tesis demuestra cómo hacerlo, cómo fomentar colaboraciones y compartición de recursos entre personas.

Nuestro entorno estadístico basado en *workflows* con Taverna, Tavlab, aspira a ser una contribución original al análisis estadístico de datos experimentales dentro de la *e-Science*.

Trabajos futuros:

1. En virtud de que nuestra infraestructura es software abierto, es factible añadir servicios Web con otras aplicaciones estadísticas o de otro tipo, por ejemplo, para algoritmos en el área de Reconocimiento de Patrones, extendiendo las implementadas para nuestro entorno Tavlab.

2. Publicación de servicios Web por medio de PeDRo. PeDRo es una herramienta abierta para la captura de datos basada en XML. PeDRo permite especificar documentos XML como el WSDL utilizado para describir servicios Web. La especificación de un documento XML se realiza por medio de la captura de los elementos y atributos más representativos del servicio Web que describen. Los elementos y atributos capturados por medio de PeDRo formarán parte de una ontología que permitirá fácilmente la localización del servicio Web requerido.

3. Creación de un motor de capa de conocimiento que pueda añadirle a los servicios Web una ontología tal que puedan ser localizados por un motor de búsqueda de servicios como Feta, lo cual constituye un reto en todos los sistemas de componentes de software cuya finalidad es ser reutilizados por otros. En la medida que un componente de software es utilizado ampliamente, éste se convierte en un estándar; a diferencia de la manufactura de software tradicional en la que cada componente tiene un objetivo específico y sólo ese.

Apéndice A

A.1 Instalación de software

I. Instalación del servidor Apache-Tomcat

Para poner en marcha el servidor Apache Tomcat, se instala el software que se describe a continuación:

1. **Instalar jdk**

   JDK (Java Development Kit) constituye el paquete Java de Sun MicroSystems y es el que nos permite compilar y ejecutar nuestros programas escritos en este lenguaje.

   Se debe contar con la versión 1.5 o mayor.

   Se instaló el paquete: **jdk-1_5_0_08-linux-i586.bin** en el directorio /usr/local

   Se agregan al directorio /etc/profile las siguientes variables de ambiente:

   export JAVA_HOME=/usr/local/jdk1.5.0_08

   export JRE_HOME=$JAVA_HOME/jre

   export PATH=$PATH:$JAVA_HOME/bin

   Recargar la sesión: salir con *logout* y entrar nuevamente con *login*.

   Ejecutar *javac* para asegurar que la instalación haya sido correcta.

2. **Instalar Apache Tomcat**

   Apache-Tomcat es el paquete con los programas que permiten levantar el servidor como contenedor para nuestros servicios Web.

   Se instaló el paquete: **apache-tomcat.5.5.17.tar.gz** en el directorio /usr/local.

   Se agregan al directorio /etc/profile la siguiente variable de ambiente:

   export CATALINA_HOME=/usr/local/apache-tomcat-5.5.17

Recargar la sesión: salir con *logout* y entrar nuevamente con *login*.

3. **Instalar xerces, jaf y javamail**

   Xerces es un analizador (*parser*) para código XML escrito en un subconjunto portable de C++ por el proyecto Apache.

   JAF es el paquete *JavaBeans Activator Framework* para la edición de documentos.

   javamail es utilizado para aplicaciones de envío de mensajes y correo electrónico independientemente de la plataforma y protocolos.

   tar -xvzf Xerces-J-bin.1.4.4.tar.gz

   mv xerces-1_4_4/ /usr/local

   unzip jaf-1_1-fr.zip

   mv jaf-1.1 /usr/local

   unzip javamail-1_4.zip

   mv javamail-1.4 /usr/local

   Agregar al /etc/profile (al final del archivo):

   export CLASSPATH=usr/local/xerces-1_4_4/xerces.jar:$PATH

   export CLASSPATH=$CLASSPATH:usr/local/javamail-1.4/mail.jar

   export CLASSPATH=$CLASSPATH:usr/local/jaf-1.1/activation.jar

4. **Instalar Axis**

   Axis es la plataforma de Java para crear y ejecutar aplicaciones de servicios Web.

   tar -xvzf axis-bin-1_4.tar.gz

   mv axis-1_4 /usr/local

   Copiar la carpeta axis al directorio /webapps de Apache-Tomcat:

   cp -a /usr/local axis-1_4/webapps/axis /usr/local/apache-tomcat-5.5.17/webapps

   Agregar al /etc/profile (al final del archivo):

export AXIS_HOME=/usr/local/axis-1_4

export AXIS_LIB=$AXIS_HOME/lib

export CLASSPATH=$AXIS_LIB/axis.jar

export CLASSPATH=$CLASSPATH:$AXIS_LIB/jaxrpc.jar

export CLASSPATH=$CLASSPATH:$AXIS_LIB/saaj.jar

export CLASSPATH=$CLASSPATH:$AXIS_LIB/commons-logging-1.0.4.jar

export CLASSPATH=$CLASSPATH:$AXIS_LIB/commons-discovery-0.2.jar

export CLASSPATH=$CLASSPATH:$AXIS_LIB/wsdl4j-1.5.1.jar

export CLASSPATH=$CLASSPATH:$AXIS_HOME/samples

export CLASSPATH=$CLASSPATH:$AXIS_LIB/xmlsec-1.3.0.jar

export CLASSPATH=$CLASSPATH:$AXIS_LIB/activation.jar

export CLASSPATH=$CLASSPATH:$AXIS_LIB/mail.jar

export CLASSPATH=$CLASSPATH:$AXIS_LIB/xml-apis.jar

export CLASSPATH=$CLASSPATH:$AXIS_LIB/xercesImpl.jar

Recargar la sesión: salir con *logout* y entrar nuevamente con *login*.

II. Instalación de Taverna

Se instaló la versión 1.5.2 de Taverna en el directorio:

/home/ulises01

En el directorio /usr (o /usr/bin según sea el caso) buscar el enlace simbólico a java.

/usr# ls -al java

lrwxrwxrwx 1 root root 22 2006-08-22 14:30 java ⟶ /etc/alternatives/java

Renombrar java a java_old porque era un enlace simbólico del java de GNU.

/usr# mv java java_old

Crear nuestro propio enlace al java de Sun.

/usr# ln -s /usr/local/jdk1.5.0_08/bin/java java

Desde /home/ulises01/taverna-1.5.2 ejecutar Taverna mediante:

./runme.sh

Instalar el software auxiliar para gráficas *graphviz* de AT&T que puede ser bajado gratuitamente de http://www.research.att.com/sw/tools/graphviz/download.html.

graphviz se instaló en en el directorio:

/home/ulises01

Desde /home/ulises01/graphviz ejecutar con:

./install.sh

De esta manera la variable de ambiente CLASSPATH quedó como muestra la *Figura A.1.*

Figura A.1: Variable de ambiente CLASSPATH

III. Probar la instalación

Para levantar el servidor **Apache-Tomcat** se procede como indica la *Figura A.2*. Luego desde la ventana de un navegador o *browser* abrimos: http://localhost:8080/.

Si se puede ver esta página (*Figura A.3*), entonces se ha instalado el servidor **Apache-Tomcat** correctamente.

Figura A.2: Arranque del servidor Apache-Tomcat

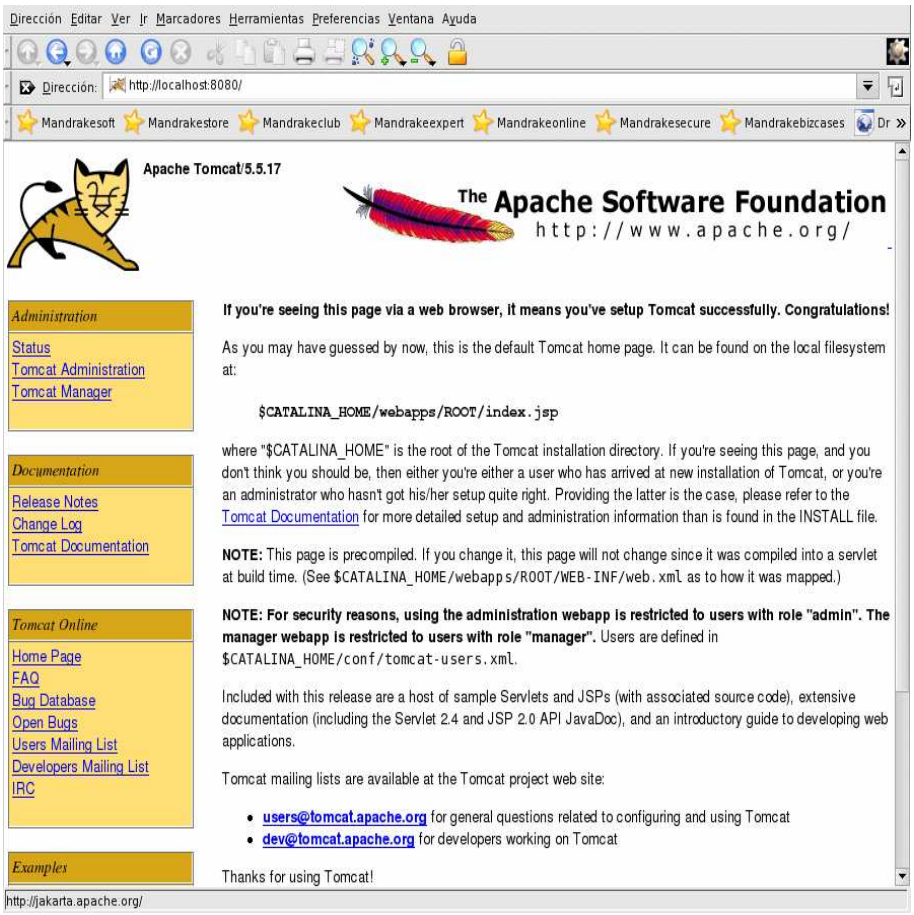


Figura A.3: Página de instalación de Apache-Tomcat

Desde este mismo navegador abrimos ahora: http://localhost:8080/axis.

Aparece la pantalla de instalación y validación del software de **Axis** instalado en el servidor **Apache-Tomcat** que se muestra en la *Figura A.4* y se selecciona la liga *Validation*.

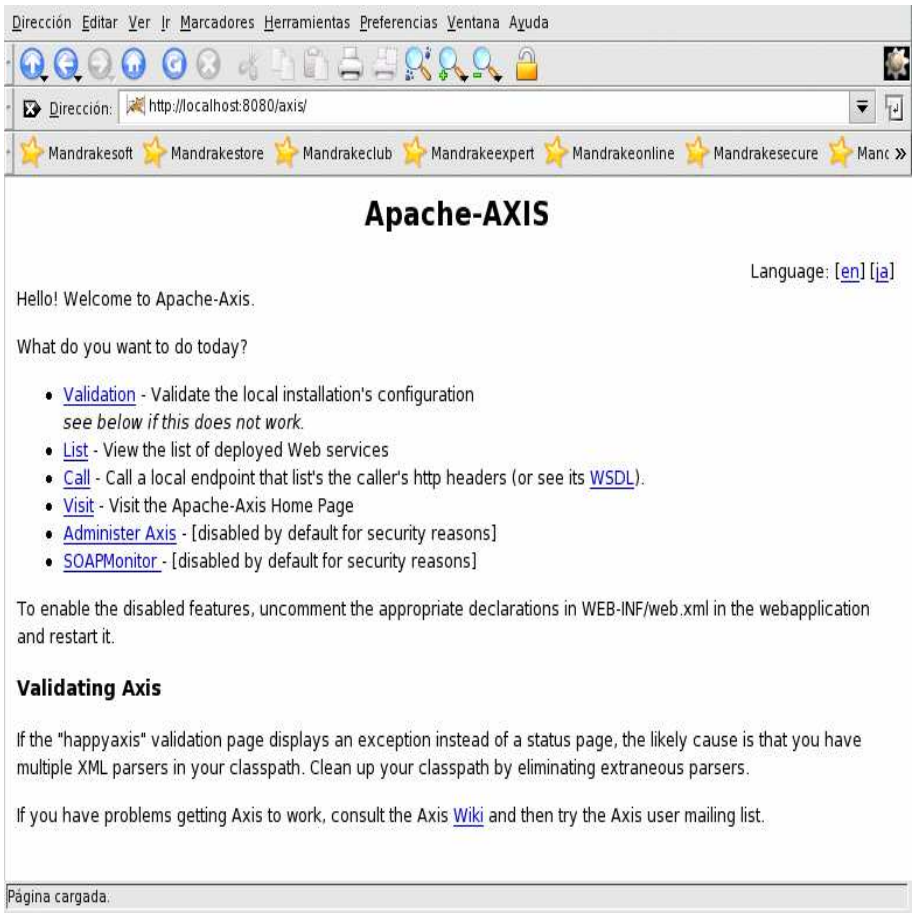


Figura A.4: Página de instalación de Axis

A continuación, aparece una nueva pantalla (*Figura A.5*) mostrando que los componentes necesarios para el funcionamiento de **Axis** fueron encontrados.

La parte inferior de esta misma pantalla (*Figura A.6*) muestra que los componentes opcionales para el funcionamiento de **Axis** fueron encontrados.

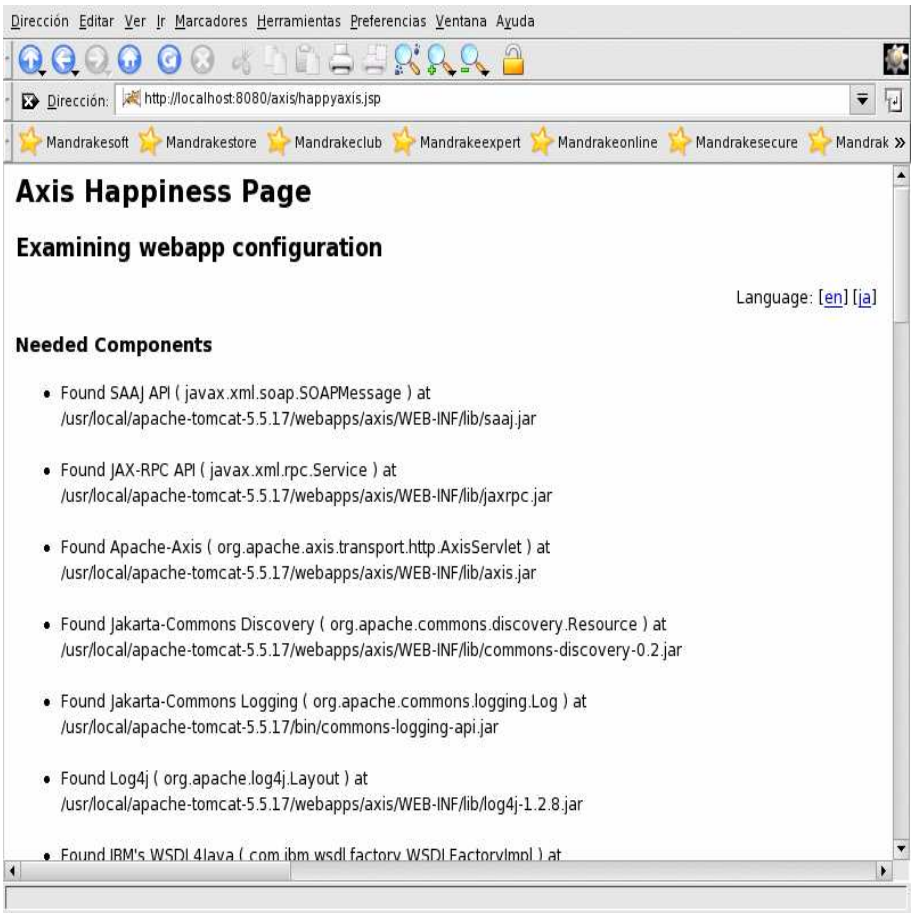


Figura A.5: Página de validación de Axis/componentes necesarios

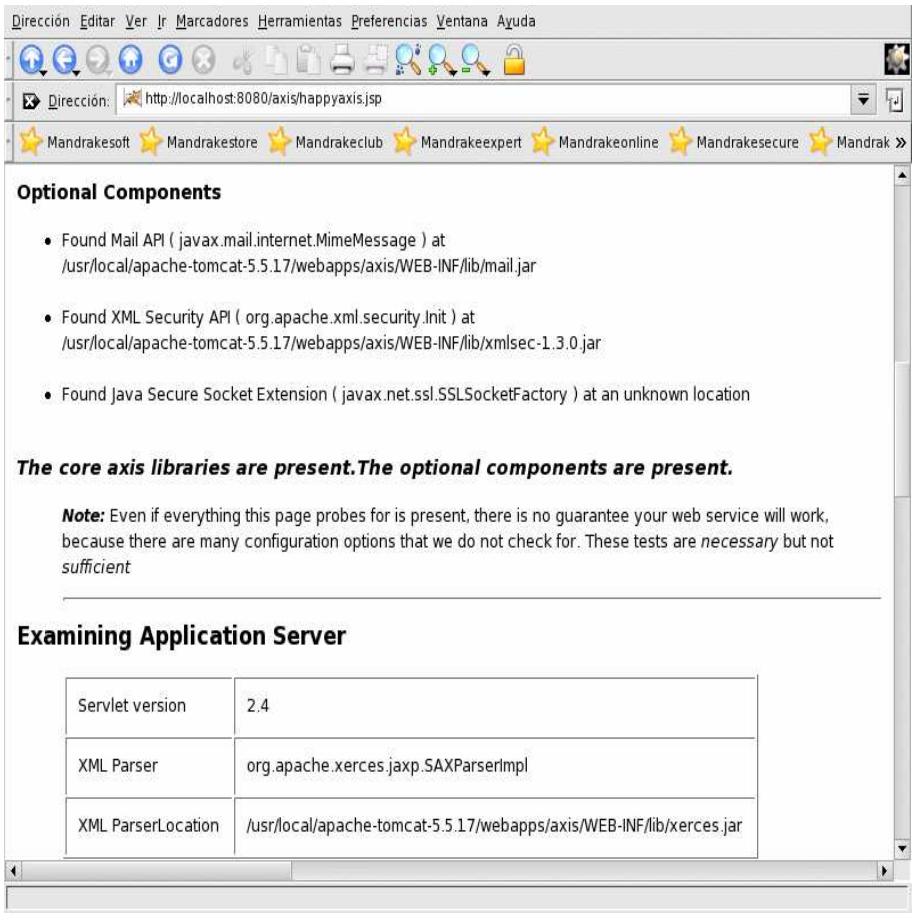


Figura A.6: Página de validación de Axis/componentes opcionales

A.2 Creación de un documento descriptor de activación de servicios Web WSDD (*Web Services Deployment Descriptor*)

a) Se compila al archivo **.java** de la aplicación con la siguiente instrucción:

javac Programa.java

b) Asegúrese de que el archivo **.class** se encuentre en el siguiente directorio:

/usr/local/apache-tomcat-5.5.17/webapps/axis/WEB-INF/classes

c) Definimos su descritor de activación o de despliegue (wsdd):

Programa.wsdd

<deployment xmlns="http://xml.apache.org/axis/wsdd/"
xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">

<!– Definimos el servicio Web a activar, nombre y tipo de servicio Web.
RPC llamadas a procedimientos remotos con ejecución síncrono >

<service name="ProgramaWS" provider="java:RPC">

<!– Nombre de la clase que implementa los métodos expuestos –>
<parameter name="className" value="Programa"/ >

<!– Expone todos los métodos como visibles desde el exterior –>
<parameter name="allowedMethods" value"*"/ >
</service>
</deployment>

d) Ejecutamos el comando:

java org.apache.axis.client.AdminClient Programa.wsdd

Si la ruta desde donde se ejecuta la activación (deploy) no se encuentra el **.wsdd**, devolverá una excepción diciendo que el sistema no pudo encontrar el archivo especificado:

java.io.FileNotFoundException: Programa.wsdd

e) Terminado esto, habrá de aparecer listado nuestro servicio con sus respectivos métodos en la dirección:

http://localhost:8080/axis/servlet/AxisServlet

Esto se muestra en la *FiguraA.7*. Al seleccionar la liga **wsdl** que aparece a un lado de cada servicio Web en la *Figura A.7* se obtendrá su documento descriptor pero sin los nombres de espacio (namespace). De esta manera, hay que buscar en Internet los posibles nombres de espacio para nuestra aplicación (*Figura A.8*).

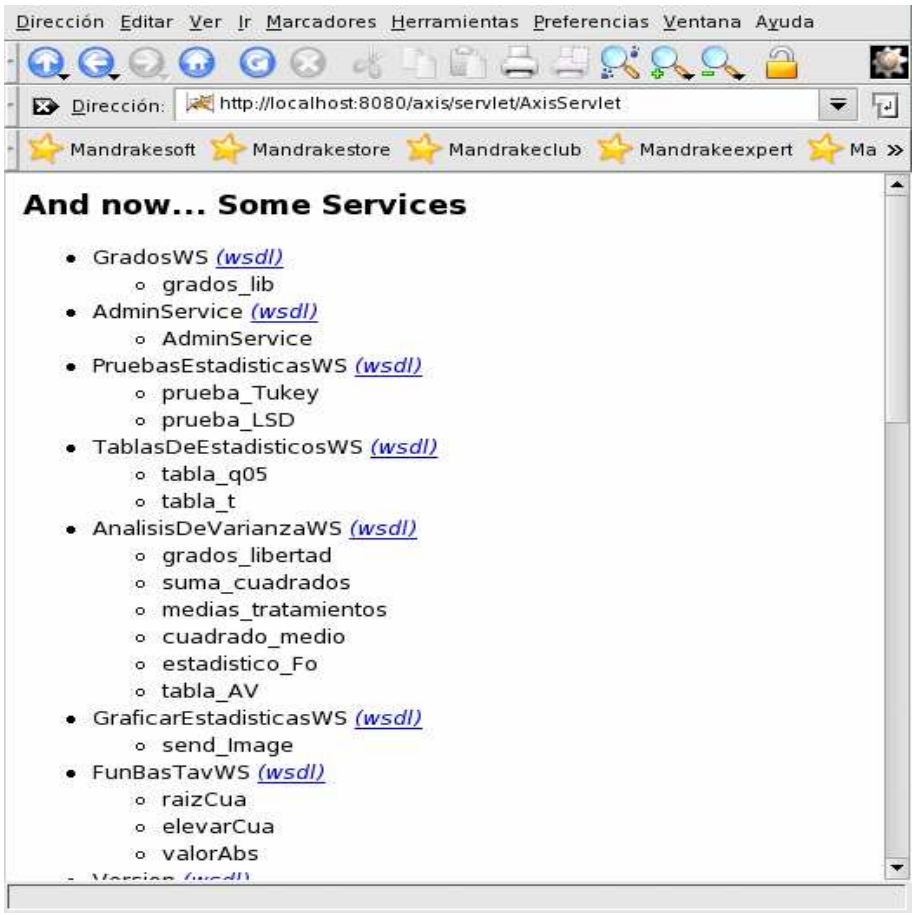


Figura A.7: Pantalla de servicios

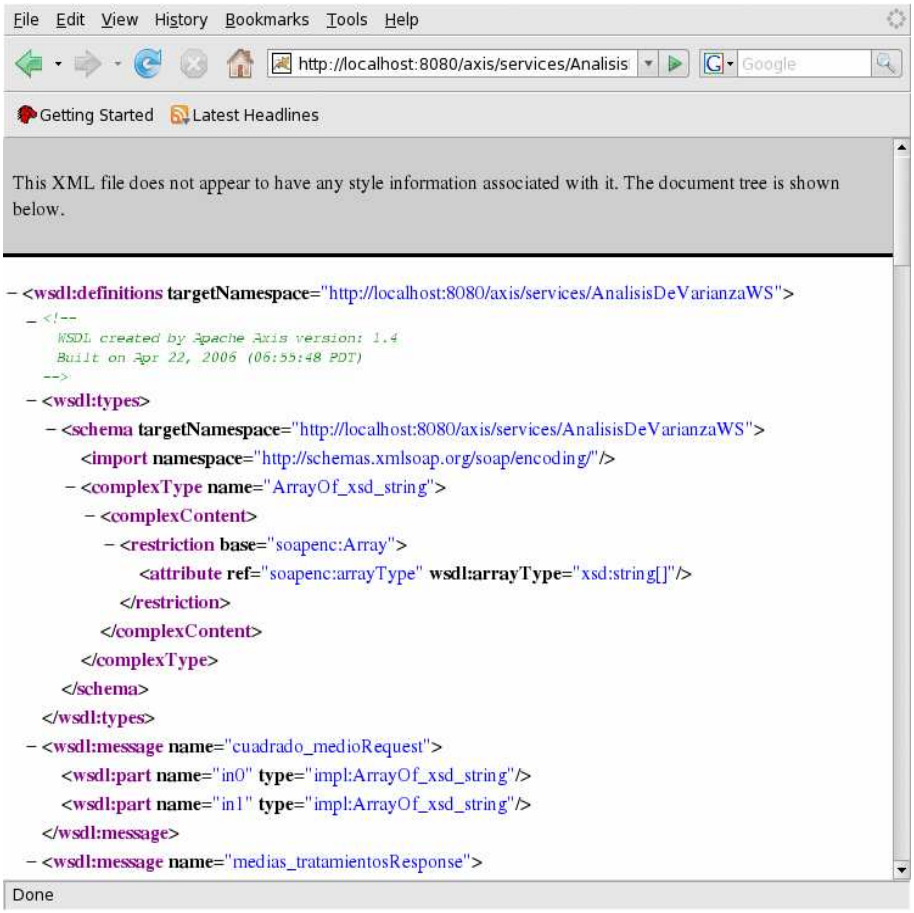


Figura A.8: Documento WSDL sin nombres de espacio (namespace)

A.3 Descripción de las conexiones de los métodos implementados

A continuación, mediante las *Tablas A.1* a *A.7*, se describen las entradas y salidas de los diferentes métodos pertenecientes a los servicios Web destinados a las aplicaciones en Biotecnología. Los íconos que representan a estos métodos se interconectan entre sí para construir los *workflows* correspondientes al Capítulo 5.

Las entradas y salidas de los íconos que representan a los métodos del servicio Web *RecibeDatosExperimentalesWS* se describen en la *Tabla A.1*.

RecibeDatosExperimentalesWS

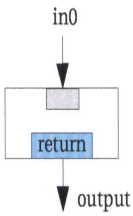
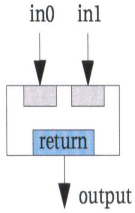
Método	Conexiones del ícono	Estructura de datos
recibe_datos		Entrada **in0:** arreglo de argumentos tipo string Salida Arreglo tipo string de datos sin líneas de retorno ni espacios
datos_tratamientos		Entrada **in0:** arreglo de argumentos tipo string **in1:** número de tratamientos Salida Arreglo tipo string de datos sin líneas de retorno ni espacios El último elemento corresponde al número de réplicas en los tratamientos

Tabla A.1: Entrada/salida de datos de los métodos de *RecibeDatosExperimentalesWS*

En las *Tablas A.2, A.3* y *A.4*, se describen las entradas y salidas de los íconos correspondientes a los métodos del servicio Web *AnalisisDeVarianzaWS*.

AnalisisDeVarianzaWS

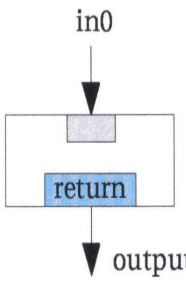
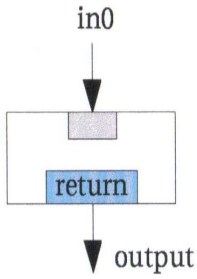
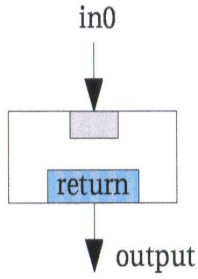
Método	Conexiones del ícono	Estructura de datos
grados_libertad		Entradas **in0:** arreglo tipo string de datos correspondiente a la salida de *datos_tratamientos* Salida Arreglo tipo string de datos
suma_cuadrados		Entradas **in0:** arreglo tipo string de datos correspondiente a la salida de *datos_tratamientos* Salida Arreglo tipo string de datos
medias_tratamientos		Entradas **in0:** arreglo tipo string de datos correspondiente a la salida de *datos_tratamientos* Salida Arreglo tipo string de datos

Tabla A.2: Entrada/salida de datos de los métodos de *AnalisisDeVarianzaWS*

AnalisisDeVarianzaWS

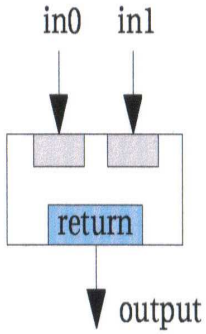
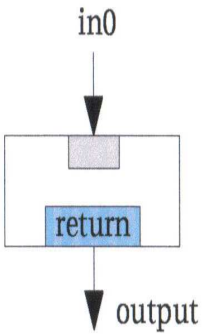
Método	Conexiones del ícono	Estructura de datos
cuadrado_medio		Entradas **in0:** arreglo tipo string de datos correspondiente a la salida de *suma_cuadrados* **in1:** arreglo tipo string de datos correspondiente a la salida de *grados_libertad* Salida Arreglo tipo string de datos
estadistico_Fo		Entradas **in0:** arreglo tipo string de datos correspondiente a la salida de *cuadrado_medio* Salida Valor tipo *double*

Tabla A.3: Entrada/salida de datos de los métodos de *AnalisisDeVarianzaWS*

AnalisisDeVarianzaWS

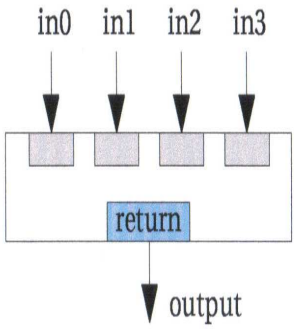
Método	Conexiones del ícono	Estructura de datos
tabla_AV		Entradas **in0:** arreglo tipo string de datos correspondiente a la salida de *suma_cuadrados* **in1:** arreglo tipo string de datos correspondiente a la salida de *grados_libertad* **in2:** arreglo tipo string de datos correspondiente a la salida de *cuadrado_medio* **in3:** valor tipo *double* correspondiente a la salida de *estadistico_Fo* Salida Arreglo tipo *byte* correspondiente a la codificación de una imagen *jpeg* en Java

Tabla A.4: Entrada/salida de datos de los métodos de *AnalisisDeVarianzaWS*

En la *Tabla A.5* se presentan las entradas y salidas para los íconos de los métodos del servicio Web *TablasDeEstadisticosWS*.

TablasDeEstadisticosWS

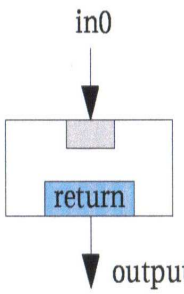
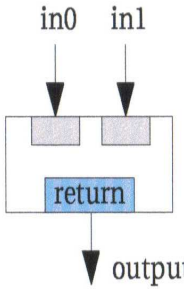
Método	Conexiones del ícono	Estructura de datos
tabla_q05		Entradas **in0:** arreglo tipo string de datos correspondiente a la salida de *datos_tratamientos* Salida Valor tipo *double* correspondiente al valor del estadístico *q* para *f* grados de libertad y *a* tratamientos con *alfa* igual a 0.05
tabla_t		Entradas **in0:** arreglo tipo string de datos correspondiente a la salida de *grados_libertad* **in1:** valor tipo *double* correspondiente al valor de *alfa* Salida Valor tipo *double* correspondiente al valor del estadístico *t* para *f* grados de libertad y *alfa*

Tabla A.5: Entrada/salida de datos de los métodos de *TablasDeEstadisticosWS*

Mediante las *Tablas A.6* y *A.7*, se describen las entradas y salidas de los íconos correspondientes a los métodos del servicio Web *PruebasEstadisticasWS*.

PruebasEstadisticasWS

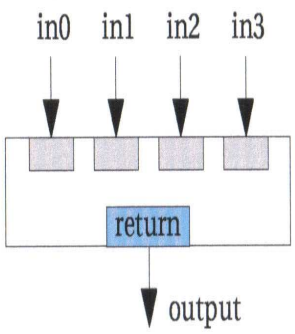
Método	Conexiones del ícono	Estructura de datos
prueba_Tukey		Entradas **in0:** arreglo tipo string de datos correspondiente a la salida de *suma_cuadrados* **in1:** arreglo tipo string de datos correspondiente a la salida de *grados_libertad* **in2:** valor tipo *double* correspondiente a la salida de *tabla_q05* **in3:** arreglo tipo string de datos correspondiente a la salida de *medias_tratamientos* Salida Arreglo tipo *byte* correspondiente a la codificación de una imagen *jpeg* en Java

Tabla A.6: Entrada/salida de datos de los métodos de *PruebasEstadisticasWS*

PruebasEstadisticasWS

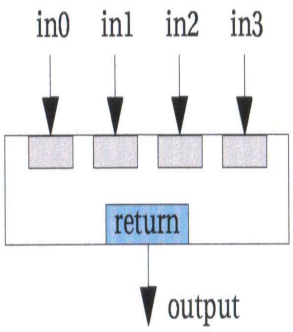
Método	Conexiones del ícono	Estructura de datos
prueba_LSD		Entradas **in0:** arreglo tipo string de datos correspondiente a la salida de *suma_cuadrados* **in1:** arreglo tipo string de datos correspondiente a la salida de *grados_libertad* **in2:** valor tipo *double* correspondiente a la salida de *tabla_t* **in3:** arreglo tipo string de datos correspondiente a la salida de *medias_tratamientos* Salida Arreglo tipo *byte* correspondiente a la codificación de una imagen *jpeg* en Java

Tabla A.7: Entrada/salida de datos de los métodos de *PruebasEstadisticasWS*

En la *Tabla A.8* se presentan las entradas y salidas para los íconos de los métodos del servicio Web *RecibeArchivosBinariosWS*. En la *Tabla A.8* sólo se muestra el método *binarioInteger_BigEndian* ya que los demás métodos de este servicio Web, tienen la misma representación.

RecibeArchivosBinariosWS

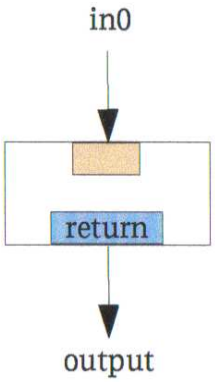
Método	Conexiones del ícono	Estructura de datos
binarioInteger_BigEndian		Entrada **in0**: archivo binario de datos Salida Arreglo de datos tipo string

Tabla A.8: Entrada/salida de datos de los métodos de *RecibeArchivosBinariosWS*

De la *Tabla A.9* a la *Tabla A.15*, se describen las entradas y salidas de los íconos correspondientes a los métodos del servicio Web *ApEstadisticasTavWS*.

ApEstadisticasTavWS

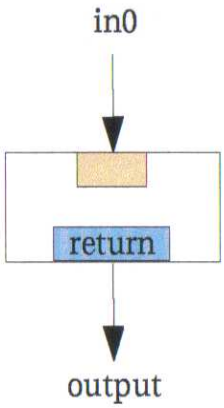
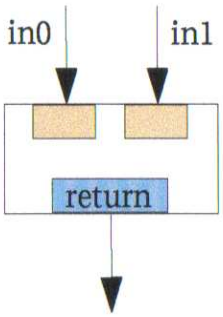
Método	Conexiones del ícono	Estructura de datos
media		Entrada **in0**: arreglo de datos tipo string correspondiente a la salida de *recibe_datos* Salida Dato tipo double
varianza		Entrada **in0**: arreglo de datos tipo string correspondiente a la salida de *recibe_datos* **in1**: dato tipo double correspondiente a la salida de *media* Salida Dato tipo double

Tabla A.9: Entrada/salida de datos de los métodos de *ApEstadisticasTavWS*

ApEstadisticasTavWS

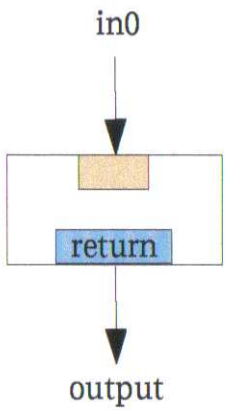
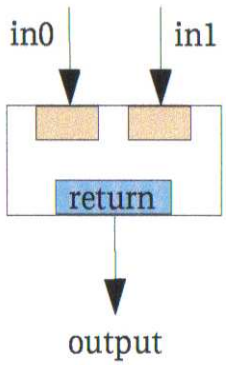
Método	Conexiones del ícono	Estructura de datos
mediana		Entrada **in0**: arreglo de datos tipo string correspondiente a la salida de *recibe_datos* Salida Dato tipo double
student_t_Corr		Entrada **in0**: arreglo de datos tipo string correspondiente a la salida de *recibe_datos* **in1**: dato tipo double correspondiente a la salida de *coeCorr* Salida Dato tipo string

Tabla A.10: Entrada/salida de datos de los métodos de *ApEstadisticasTavWS*

ApEstadisticasTavWS

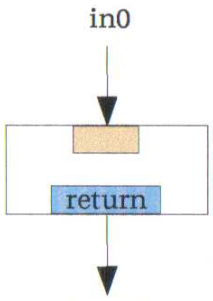
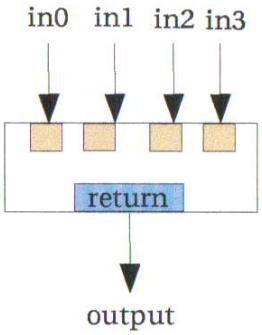
Método	Conexiones del ícono	Estructura de datos
moda		Entrada **in0**: arreglo de datos tipo string correspondiente a la salida de *recibe_datos* Salida Dato tipo double
var_Conjunta		Entrada **in0**: arreglo de datos tipo string correspondiente a la salida de *recibe_datos* **in1**: arreglo de datos tipo string correspondiente a la salida de *recibe_datos1* **in2**: dato tipo double correspondiente a la salida de la *varianza* **int3**: dato tipo double correspondiente a la salida de la *varianza1* Salida Arreglo de datos tipo string

Tabla A.11: Entrada/salida de datos de los métodos de *ApEstadisticasTavWS*

ApEstadisticasTavWS

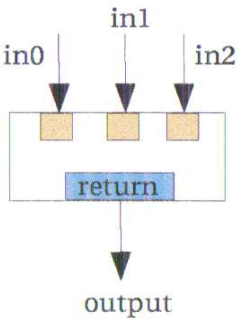
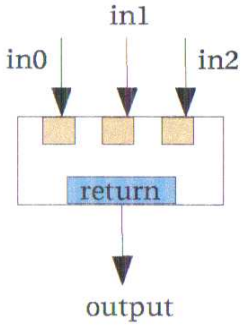
Método	Conexiones del ícono	Estructura de datos
ecuReg		Entrada **in0**: dato tipo double correspondiente a la salida de *coeReg* **in1**: dato tipo double correspondiente a la salida de *media* **in2**: dato tipo double correspondiente a la salida de *media1* Salida Arreglo de datos tipo string
student_t_Vc		Entrada **in0**: arreglo de datos tipo string correspondiente a la salida de *var_Conjunta* **in1**: dato tipo double correspondiente a la salida de *media* **in2**: dato tipo double correspondiente a la salida de *media1* Salida Dato tipo string

Tabla A.12: Entrada/salida de datos de los métodos de *ApEstadisticasTavWS*

ApEstadisticasTavWS

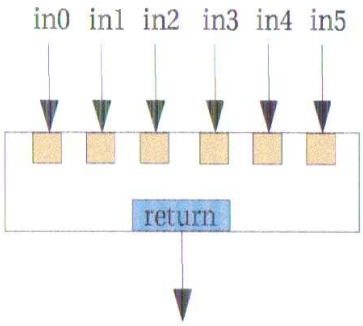
Método	Conexiones del ícono	Estructura de datos
coeCorr		Entrada **in0**: arreglo de datos tipo string correspondiente a la salida de *recibe_datos* **in1**: arreglo de datos tipo string correspondiente a la salida de *recibe_datos1* **int2**: dato tipo double correspondiente a la salida de *media* **int3**: dato tipo double correspondiente a la salida de *media1* **int4**: dato tipo double correspondiente a la desviación estándar (*varianza -> raizCua)* **int5**: dato tipo double correspondiente a la desviación estándar1 (*varianza1 -> raizCua1)* Salida Dato tipo double

Tabla A.13: Entrada/salida de datos de los métodos de *ApEstadisticasTavWS*

ApEstadisticasTavWS

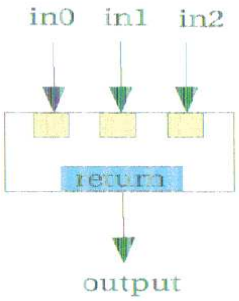
Método	Conexiones del ícono	Estructura de datos
coeReg		Entrada **in0**: dato tipo double correspondiente a la salida de *coeCorr* **in1**: dato tipo double correspondiente a la desviación estándar **in2**: dato tipo double correspondiente a la desviación estándar1 (*varianza1 -> raizCua1*) Salida Dato tipo double

Tabla A.14: Entrada/salida de datos de los métodos de *ApEstadisticasTavWS*

ApEstadisticasTavWS

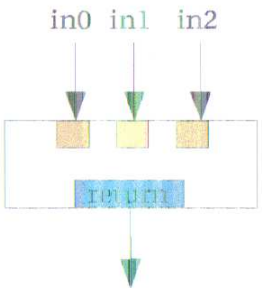
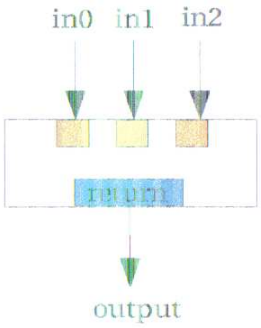
Método	Conexiones del ícono	Estructura de datos
desvResEst		Entrada **in0**: arreglo de datos tipo string correspondiente a la salida de *recibe_datos* **in2**: dato tipo double correspondiente a la salida de *varianza1* **int3**: dato tipo double correspondiente al cuadrado (*elevarCua*) de la salida de *coeCorr* Salida Dato tipo double
errorEst_m		Entrada **in0**: arreglo de datos tipo string correspondiente a la salida de *recibe_datos* **in1**: dato tipo double correspondiente a la salida de *media* **in2**: dato tipo double correspondiente a la salida de *desvResEst* Salida Dato tipo double

Tabla A.15: Entrada/salida de datos de los métodos de *ApEstadisticasTavWS*

En la *Tabla A.16* se presentan las entradas y salidas para los íconos de los métodos del servicio Web *FunBasTavWS*.

FunBasTavWS

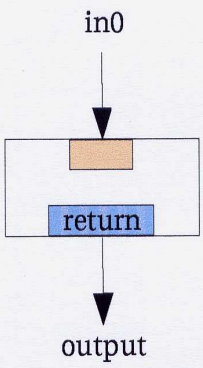
Método	Conexiones del ícono	Estructura de datos
raizCua *elevarCua* *valorAbs*		Entrada **int0**: dato tipo double Salida Dato tipo double

Tabla A.16: Entrada/salida de datos de los métodos de *FunBasTavWS*

En la *Tabla A.17* se presentan las entradas y salidas para el ícono correspondiente al método del servicio Web *GraficadorTavWS*.

GraficadorTavWS

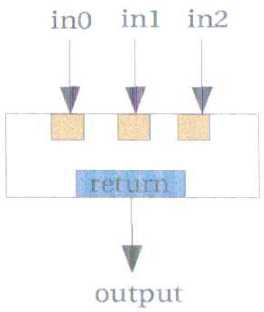
Método	Conexiones del ícono	Estructura de datos
graficar_ER		Entrada **in0**: arreglo de datos tipo string correspondiente a la salida de *recibe_datos* **in1**: arreglo de datos tipo string correspondiente a la salida de *recibe_datos1* **int2**: arreglo de datos tipo string correspondiente a la salida de *ecuReg* Salida Arreglo tipo byte

Tabla A.17: Entrada/salida de datos de los métodos de *GraficadorTavWS*

A.4 SCUFL y BPEL4WS

Taverna utiliza el lenguaje de flujo unificado simple conceptual SCUFL (Simple Conceptual Unified Flow Language), basado en XML, para describir *workflows* y localizar los servicios Web mediante sus respectivas URLs. SCUFL coordina la ejecución del *workflow*.

Taverna utiliza SCUFL en lugar del estándar BPEL4WS (Business Process Execution Language for Web Services) porque es más apropiado para representar *workflows* científicos. BPEL4WS está enfocado a representar y coordinar *workflows* de procesos de negocios. Utilizar BPEL4WS disminuiría las principales funcionalidades de los *workflows* científicos (como son las características de diagnóstico) [83].

Aunque es factible realizar una conversión de SCUFL a BPEL4WS esto requiriría de mucho esfuerzo. Es más, el esfuerzo sería aún mayor una vez que aparezca disponible Taverna2 ya que el distanciamiento entre las 2 se incrementa.

A continuación, se muestra una sección de código SCUFL (*Figura A.9*) que describe al *workflow* que se muestra a su derecha.

SCUFL (*Simple Conceptual Unified Flow Language*)

```
<s:processor name ="media">
  <s:arbitrarywsdl>
    <s:wsdl>http://tavlab.cs.cinvestav.mx:2112/axis/services/ApEstadisticasTavWS?wsdl</s:wsdl>
    <s:operation>media</s:operation>
  </s:arbitrarywsdl>
</s:processor>
<s:processor name ="varianza">
  <s:arbitrarywsdl>
    <s:wsdl>http://tavlab.cs.cinvestav.mx:2112/axis/services/ApEstadisticasTavWS
    <s:operation>varianza</s:operation>
  </s:arbitrarywsdl>
</s:processor>
<s:link source="Valoraciones" sink="recibe_datos: in0" />
<s:link source="media:mediaReturn" sink="media" />
<s:link source="media:mediaReturn" sink="varianza: in1" />
<s:link source="raizCua:raizCuaReturn" sink="desvEst" />
<s:link source="recibe_datos:recibe_datosReturn" sink="media: in0" />
<s:link source="recibe_datos:recibe_datosReturn" sink="varianza: in0" />
<s:link source="varianza_varianzaReturn" sink="raizCua: in0" />
<s:source name:"Valoraciones" />
<s:sink name="media" />
<s:sink name="desvEst" />
```

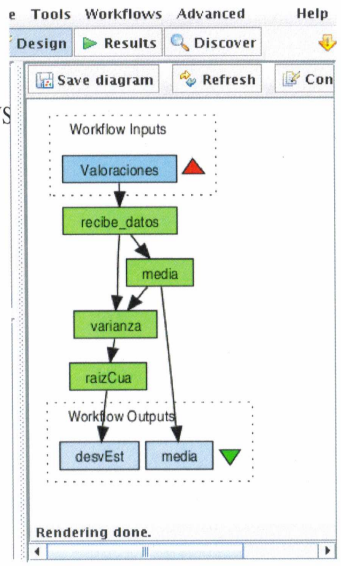


Figura A.9: Sección de código SCUFL

A.5 Participación en el *Taverna users list*

Durante nuestra investigación detectamos y reportamos un *bug* o falla en Taverna al momento de introducir datos manualmente (que no provienen de un archivo) para alimentar nuestros *workflows*. El correo se muestra en la *Figura A.10*.

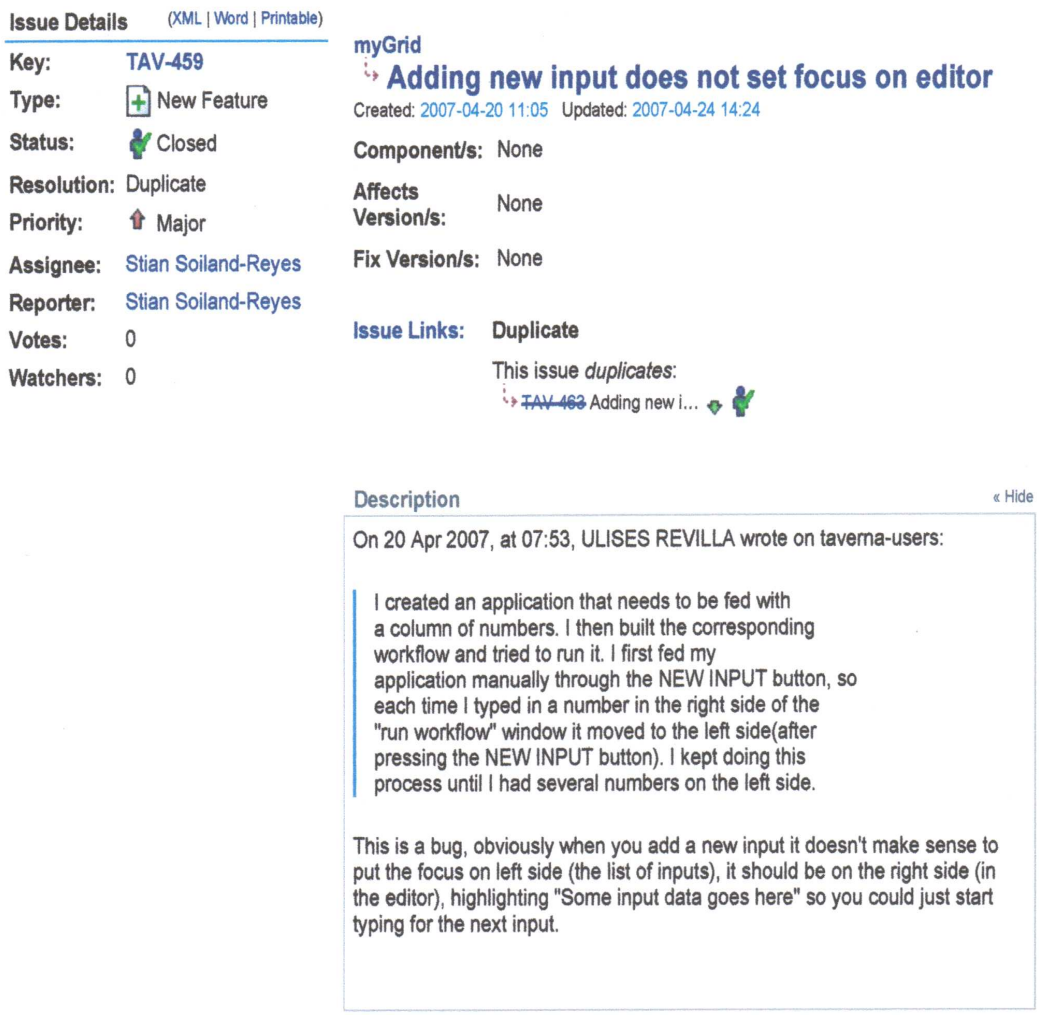


Figura A.10: Participación en el *Taverna users list*

Apéndice B

B.1 Tablas estadísticas utilizadas

En este apéndice se incluyen las tablas estadísticas utilizadas en el Capítulo 5. Las *Tablas B.1 y B.2*, corresponden a los puntos porcentuales de la distribución F y la distribución t respectivamente. Y la *Tabla B.3* corresponde a los puntos porcentuales del estadístico de rango studentizado q.

IV. Puntos porcentuales de la distribución F (*continuación*)

$F_{0.05,\nu_1,\nu_2}$

ν_2 \ ν_1	Grados de libertad del numerador (ν_1)																		
	1	2	3	4	5	6	7	8	9	10	12	15	20	24	30	40	60	120	∞
	161.4	199.5	215.7	224.6	230.2	234.0	236.8	238.9	240.5	241.9	243.9	245.9	248.0	249.1	250.1	251.1	252.2	253.3	254.3
2	18.51	19.00	19.16	19.25	19.30	19.33	19.35	19.37	19.38	19.40	19.41	19.43	19.45	19.45	19.46	19.47	19.48	19.49	19.50
3	10.13	9.55	9.28	9.12	9.01	8.94	8.89	8.85	8.81	8.79	8.74	8.70	8.66	8.64	8.62	8.59	8.57	8.55	8.53
4	7.71	6.94	6.59	6.39	6.26	6.16	6.09	6.04	6.00	5.96	5.91	5.86	5.80	5.77	5.75	5.72	5.69	5.66	5.63
5	6.61	5.79	5.41	5.19	5.05	4.95	4.88	4.82	4.77	4.74	4.68	4.62	4.56	4.53	4.50	4.46	4.43	4.40	4.36
6	5.99	5.14	4.76	4.53	4.39	4.28	4.21	4.15	4.10	4.06	4.00	3.94	3.87	3.84	3.81	3.77	3.74	3.70	3.67
7	5.59	4.74	4.35	4.12	3.97	3.87	3.79	3.73	3.68	3.64	3.57	3.51	3.44	3.41	3.38	3.34	3.30	3.27	3.23
8	5.32	4.46	4.07	3.84	3.69	3.58	3.50	3.44	3.39	3.35	3.28	3.22	3.15	3.12	3.08	3.04	3.01	2.97	2.93
9	5.12	4.26	3.86	3.63	3.48	3.37	3.29	3.23	3.18	3.14	3.07	3.01	2.94	2.90	2.86	2.83	2.79	2.75	2.71
10	4.96	4.10	3.71	3.48	3.33	3.22	3.14	3.07	3.02	2.98	2.91	2.85	2.77	2.74	2.70	2.66	2.62	2.58	2.54
11	4.84	3.98	3.59	3.36	3.20	3.09	3.01	2.95	2.90	2.85	2.79	2.72	2.65	2.61	2.57	2.53	2.49	2.45	2.40
12	4.75	3.89	3.49	3.26	3.11	3.00	2.91	2.85	2.80	2.75	2.69	2.62	2.54	2.51	2.47	2.43	2.38	2.34	2.30
13	4.67	3.81	3.41	3.18	3.03	2.92	2.83	2.77	2.71	2.67	2.60	2.53	2.46	2.42	2.38	2.34	2.30	2.25	2.21
14	4.60	3.74	3.34	3.11	2.96	2.85	2.76	2.70	2.65	2.60	2.53	2.46	2.39	2.35	2.31	2.27	2.22	2.18	2.13
15	4.54	3.68	3.29	3.06	2.90	2.79	2.71	2.64	2.59	2.54	2.48	2.40	2.33	2.29	2.25	2.20	2.16	2.11	2.07
16	4.49	3.63	3.24	3.01	2.85	2.74	2.66	2.59	2.54	2.49	2.42	2.35	2.28	2.24	2.19	2.15	2.11	2.06	2.01
17	4.45	3.59	3.20	2.96	2.81	2.70	2.61	2.55	2.49	2.45	2.38	2.31	2.23	2.19	2.15	2.10	2.06	2.01	1.96
18	4.41	3.55	3.16	2.93	2.77	2.66	2.58	2.51	2.46	2.41	2.34	2.27	2.19	2.15	2.11	2.06	2.02	1.97	1.92
19	4.38	3.52	3.13	2.90	2.74	2.63	2.54	2.48	2.42	2.38	2.31	2.23	2.16	2.11	2.07	2.03	1.98	1.93	1.88
20	4.35	3.49	3.10	2.87	2.71	2.60	2.51	2.45	2.39	2.35	2.28	2.20	2.12	2.08	2.04	1.99	1.95	1.90	1.84
21	4.32	3.47	3.07	2.84	2.68	2.57	2.49	2.42	2.37	2.32	2.25	2.18	2.10	2.05	2.01	1.96	1.92	1.87	1.81
22	4.30	3.44	3.05	2.82	2.66	2.55	2.46	2.40	2.34	2.30	2.23	2.15	2.07	2.03	1.98	1.94	1.89	1.84	1.78
23	4.28	3.42	3.03	2.80	2.64	2.53	2.44	2.37	2.32	2.27	2.20	2.13	2.05	2.01	1.96	1.91	1.86	1.81	1.76
24	4.26	3.40	3.01	2.78	2.62	2.51	2.42	2.36	2.30	2.25	2.18	2.11	2.03	1.98	1.94	1.89	1.84	1.79	1.73
25	4.24	3.39	2.99	2.76	2.60	2.49	2.40	2.34	2.28	2.24	2.16	2.09	2.01	1.96	1.92	1.87	1.82	1.77	1.71
26	4.23	3.37	2.98	2.74	2.59	2.47	2.39	2.32	2.27	2.22	2.15	2.07	1.99	1.95	1.90	1.85	1.80	1.75	1.69
27	4.21	3.35	2.96	2.73	2.57	2.46	2.37	2.31	2.25	2.20	2.13	2.06	1.97	1.93	1.88	1.84	1.79	1.73	1.67
28	4.20	3.34	2.95	2.71	2.56	2.45	2.36	2.29	2.24	2.19	2.12	2.04	1.96	1.91	1.87	1.82	1.77	1.71	1.65
29	4.18	3.33	2.93	2.70	2.55	2.43	2.35	2.28	2.22	2.18	2.10	2.03	1.94	1.90	1.85	1.81	1.75	1.70	1.64
30	4.17	3.32	2.92	2.69	2.53	2.42	2.33	2.27	2.21	2.16	2.09	2.01	1.93	1.89	1.84	1.79	1.74	1.68	1.62
40	4.08	3.23	2.84	2.61	2.45	2.34	2.25	2.18	2.12	2.08	2.00	1.92	1.84	1.79	1.74	1.69	1.64	1.58	1.51
60	4.00	3.15	2.76	2.53	2.37	2.25	2.17	2.10	2.04	1.99	1.92	1.84	1.75	1.70	1.65	1.59	1.53	1.47	1.39
120	3.92	3.07	2.68	2.45	2.29	2.17	2.09	2.02	1.96	1.91	1.83	1.75	1.66	1.61	1.55	1.55	1.43	1.35	1.25
∞	3.84	3.00	2.60	2.37	2.21	2.10	2.01	1.94	1.88	1.83	1.75	1.67	1.57	1.52	1.46	1.39	1.32	1.22	1.00

Grados de libertad del denominador (ν_2)

Tabla B.1: Puntos porcentuales de la distribución F

II. Puntos porcentuales de la distribución t^a

ν \ α	.40	.25	.10	.05	.025	.01	.005	.0025	.001	.0005
1	.325	1.000	3.078	6.314	12.706	31.821	63.657	127.32	318.31	636.62
2	.289	.816	1.886	2.920	4.303	6.965	9.925	14.089	23.326	31.598
3	.277	.765	1.638	2.353	3.182	4.541	5.841	7.453	10.213	12.924
4	.271	.741	1.533	2.132	2.776	3.747	4.604	5.598	7.173	8.610
5	.267	.727	1.476	2.015	2.571	3.365	4.032	4.773	5.893	6.869
6	.265	.727	1.440	1.943	2.447	3.143	3.707	4.317	5.208	5.959
7	.263	.711	1.415	1.895	2.365	2.998	3.499	4.019	4.785	5.408
8	.262	.706	1.397	1.860	2.306	2.896	3.355	3.833	4.501	5.041
9	.261	.703	1.383	1.833	2.262	2.821	3.250	3.690	4.297	4.781
10	.260	.700	1.372	1.812	2.228	2.764	3.169	3.581	4.144	4.587
11	.260	.697	1.363	1.796	2.201	2.718	3.106	3.497	4.025	4.437
12	.259	.695	1.356	1.782	2.179	2.681	3.055	3.428	3.930	4.318
13	.259	.694	1.350	1.771	2.160	2.650	3.012	3.372	3.852	4.221
14	.258	.692	1.345	1.761	2.145	2.624	2.977	3.326	3.787	4.140
15	.258	.691	1.341	1.753	2.131	2.602	2.947	3.286	3.733	4.073
16	.258	.690	1.337	1.746	2.120	2.583	2.921	3.252	3.686	4.015
17	.257	.689	1.333	1.740	2.110	2.567	2.898	3.222	3.646	3.965
18	.257	.688	1.330	1.734	2.101	2.552	2.878	3.197	3.610	3.922
19	.257	.688	1.328	1.729	2.093	2.539	2.861	3.174	3.579	3.883
20	.257	.687	1.325	1.725	2.086	2.528	2.845	3.153	3.552	3.850
21	.257	.686	1.323	1.721	2.080	2.518	2.831	3.135	3.527	3.819
22	.256	.686	1.321	1.717	2.074	2.508	2.819	3.119	3.505	3.792
23	.256	.685	1.319	1.714	2.069	2.500	2.807	3.104	3.485	3.767
24	.256	.685	1.318	1.711	2.064	2.492	2.797	3.091	3.467	3.745
25	.256	.684	1.316	1.708	2.060	2.485	2.787	3.078	3.450	3.725
26	.256	.684	1.315	1.706	2.056	2.479	2.779	3.067	3.435	3.707
27	.256	.684	1.314	1.703	2.052	2.473	2.771	3.057	3.421	3.690
28	.256	.683	1.313	1.701	2.048	2.467	2.763	3.047	3.408	3.674
29	.256	.683	1.311	1.699	2.045	2.462	2.756	3.038	3.396	3.659
30	.256	.683	1.310	1.697	2.042	2.457	2.750	3.030	3.385	3.646
40	.255	.681	1.303	1.684	2.021	2.423	2.704	2.971	3.307	3.551
60	.254	.679	1.296	1.671	2.000	2.390	2.660	2.915	3.232	3.460
120	.254	.677	1.289	1.658	1.980	2.358	2.617	2.860	3.160	3.373
∞	.253	.674	1.282	1.645	1.960	2.326	2.576	2.807	3.090	3.291

ν = grados de libertad

Tabla B.2: Puntos porcentuales de la distribución t

VIII. Puntos porcentuales del estadístico del rango studentizado (*continuación*)

$q_{.05}(p, f)$

	p																		
f	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	18.1	26.7	32.8	37.2	40.5	43.1	45.4	47.3	49.1	50.6	51.9	53.2	54.3	55.4	56.3	57.2	58.0	58.8	59.6
2	6.09	8.28	9.80	10.89	11.73	12.43	13.03	13.54	13.99	14.39	14.75	15.08	15.38	15.65	15.91	16.14	16.36	16.57	16.77
3	4.50	5.88	6.83	7.51	8.04	8.47	8.85	9.18	9.46	9.72	9.95	10.16	10.35	10.52	10.69	10.84	10.98	11.12	11.24
4	3.93	5.00	5.76	6.31	6.73	7.06	7.35	7.60	7.83	8.03	8.21	8.37	8.52	8.67	8.80	8.92	9.03	9.14	9.24
5	3.64	4.60	5.22	5.67	6.03	6.33	6.58	6.80	6.99	7.17	7.32	7.47	7.60	7.72	7.83	7.93	8.03	8.12	8.21
6	3.46	4.34	4.90	5.31	5.63	5.89	6.12	6.32	6.49	6.65	6.79	6.92	7.04	7.14	7.24	7.34	7.43	7.51	7.59
7	3.34	4.16	4.68	5.06	5.35	5.59	5.80	5.99	6.15	6.29	6.42	6.54	6.65	6.75	6.84	6.93	7.01	7.08	7.16
8	3.26	4.04	4.53	4.89	5.17	5.40	5.60	5.77	5.92	6.05	6.18	6.29	6.39	6.48	6.57	6.65	6.73	6.80	6.87
9	3.20	3.95	4.42	4.76	5.02	5.24	5.43	5.60	5.74	5.87	5.98	6.09	6.19	6.28	6.36	6.44	6.51	6.58	6.65
10	3.15	3.88	4.33	4.66	4.91	5.12	5.30	5.46	5.60	5.72	5.83	5.93	6.03	6.12	6.20	6.27	6.34	6.41	6.47
11	3.11	3.82	4.26	4.58	4.82	5.03	5.20	5.35	5.49	5.61	5.71	5.81	5.90	5.98	6.06	6.14	6.20	6.27	6.33
12	3.08	3.77	4.20	4.51	4.75	4.95	5.12	5.27	5.40	5.51	5.61	5.71	5.80	5.88	5.95	6.02	6.09	6.15	6.21
13	3.06	3.73	4.15	4.46	4.69	4.88	5.05	5.19	5.32	5.43	5.53	5.63	5.71	5.79	5.86	5.93	6.00	6.06	6.11
14	3.03	3.70	4.11	4.41	4.64	4.83	4.99	5.13	5.25	5.36	5.46	5.56	5.64	5.72	5.79	5.86	5.92	5.98	6.03
15	3.01	3.67	4.08	4.37	4.59	4.78	4.94	5.08	5.20	5.31	5.40	5.49	5.57	5.65	5.72	5.79	5.85	5.91	5.96
16	3.00	3.65	4.05	4.34	4.56	4.74	4.90	5.03	5.15	5.26	5.35	5.44	5.52	5.59	5.66	5.73	5.79	5.84	5.90
17	2.98	3.62	4.02	4.31	4.52	4.70	4.86	4.99	5.11	5.21	5.31	5.39	5.47	5.55	5.61	5.68	5.74	5.79	5.84
18	2.97	3.61	4.00	4.28	4.49	4.67	4.83	4.96	5.07	5.17	5.27	5.35	5.43	5.50	5.57	5.63	5.69	5.74	5.79
19	2.96	3.59	3.98	4.26	4.47	4.64	4.79	4.92	5.04	5.14	5.23	5.32	5.39	5.46	5.53	5.59	5.65	5.70	5.75
20	2.95	3.58	3.96	4.24	4.45	4.62	4.77	4.90	5.01	5.11	5.20	5.28	5.36	5.43	5.50	5.56	5.61	5.66	5.71
24	2.92	3.53	3.90	4.17	4.37	4.54	4.68	4.81	4.92	5.01	5.10	5.18	5.25	5.32	5.38	5.44	5.50	5.55	5.59
30	2.89	3.48	3.84	4.11	4.30	4.46	4.60	4.72	4.83	4.92	5.00	5.08	5.15	5.21	5.27	5.33	5.38	5.43	5.48
40	2.86	3.44	3.79	4.04	4.23	4.39	4.52	4.63	4.74	4.82	4.90	4.98	5.05	5.11	5.17	5.22	5.27	5.32	5.36
60	2.83	3.40	3.74	3.98	4.16	4.31	4.44	4.55	4.65	4.73	4.81	4.88	4.94	5.00	5.06	5.11	5.15	5.20	5.24
120	2.80	3.36	3.69	3.92	4.10	4.24	4.36	4.47	4.56	4.64	4.71	4.78	4.84	4.90	4.95	5.00	5.04	5.09	5.13
∞	2.77	3.32	3.63	3.86	4.03	4.17	4.29	4.39	4.47	4.55	4.62	4.68	4.74	4.80	4.84	4.98	4.93	4.97	5.01

Tabla B.3: Puntos porcentuales del estadístico de rango studentizado q

Referencias

1. K. Wolstencroft, T. Oinn, C. Goble, J. Ferris, C. Wroe, P. Lord, K. Glover and R. Stevens, *Panoply of Utilities in Taverna*, Fist International Conference on e-Science and Grid Computing, IEEE, Volume. Issue, 5-8 Dec.2005.

2. D. Hull, K. Wolstencroft, R. Stevens, C. Globle, M. R. Pocock, P. Li and T. Oinn, *Taverna: a tool for buiding and running workflows of services*, Nucleic Acids Research, Vol. 34, Web Server issue, pp. 729-732, 2006.

3. T. Oinn, M. Greenwood, M. Addis, M. N. Alpdemir, J. Ferris, K. Glover, C. Goble, A. Goderis, D. Hull, D. Marvin P. Li, P. Lord, M. R. Pocock, M. Senger, R. Stevens, A. Wipat and C. Wroe, *Taverna: Lessons in creating a workflow environment for the life sciences*, Concurrency and Computation: Practice and Experience 2000 00:17 2005.

4. J. Zhao, P. Lord, P. Alper, C. Wroe and C. Goble, *The implications of Semantic Web technologies for support of the e-Science process*, School of Computer Science, University of Manchester, issued without date.

5. P. Lord, P. Alper, C. Wroe, R.Stevens, C. Goble, J.Zhao, D.Hull and M. Greenwood, *The Semantic Web: Service discovery and provenance in myGrid* , Department of Computer Science, University of Manchester, september 2004.

6. K. L. Garwood, C. F. Taylor, K. J. Runte, A. Brass, S. G. Oliver and N. W. Paton, *Pedro: a configurable data entry tool for XML*, Bioinformatics, Vol. 20, No. 15, pp. 2463-2465, 2004.

7. T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, M. Greenwood, T. Carver, K. Glover, M. R. Pocock, A. Wipat and P. Li, *Taverna: a tool for the composition and enactment of bioinformatics workflows*, Bioinformatics, Vol. 20, No. 17, pp. 3045-3054, 2004.

8. T. Ian, S. Matthew, W. Ian and R. Omer, *Triana Applications within Grid Computing and Peer to Peer Environments*, Journal of Grid Computing 1: 199-217, 2003.

9. A. Ilkay, B. Chad, J. Efrat, J. Matthew, L. Bertram and M. Steve, *Kepler: An Extensible System for Design and Execution of Scientific Workflows*, 14th

Intl. Conference of Scientific Workflows and Statistical Data Base Management, Santorini Island, Greece, June 2004.

10. M. Weber, T. Illman, *Using Java for the Coordination of Workflows in the World Wide Web*, Universiät Ulm, Fakultäf für Informatik, Germany, issued without a date.

11. G. M. O'Hare and N. R. Jennigs, editors, *Foundations of Distributed Artificial Intelligent*. John Wiley and Sons, 1996.

12. N. M. Avouris and L. Gasser, editors, *Distributed Artificial Intelligent: Theory and Praxis*, vol.5 of Computer and Information Science, Kluwer Academic Publishers, Boston, MA, 1992.

13. A. H. Bond and L. Gasser, *An Analysis of problems and research in DAI*, In A. H. Bond and L. Gasser, editors, Readings in Distributed Artificial Intelligence, Springer-Verlag: Heidelberg, Germany, 1988.

14. V. R. Lesser, *A restrospective view of FAC/C distributed problem solving*, IEE Transactions on Computers, C-29 (12), 1980.

15. R. B. Wesson, F. A. Hayes-Roth, J. W. Burge, C. Stasz and C. A. Sunshine, *Network structures for distributed situation assessment*, IEEE Transactions on Systems, Man and Cybernetics, C-11 (1), 1981.

16. E. Werner, *Cooperating agents: A unified theory of communication and social structure*, In L. Gasser and M. Huhns, editors, Distributed Artificial Intelligence, Volume II, Pitman Publishing: London and Morgan Kaufmann, San Mateo, CA, 1989.

17. H. Ménager and Z. Lacroix, *A workflow engine for the execution of scientific protocols*, Scientific Data management Lab, Arizona State University, USA, issued without a date.

18. S. Genaud, *Report on: The Second IEEE International Conference on e-Science* and Grid Computing, LSIIT-ICPS, Université Luis pasteur, June 7, 2007.

19. K. Schuchardr, B. Didier, G, Black, *Ecce-A Problem Solving Environment's Evolution Toward Grid Services and a Web architecture*, Pacific Northwest National Laboratory, issued without a date.

20. V. Valdivieso, *La Medicina Clínica: Una Visión Personal*, Revista chilena de pediatría, Vol. 75, No. 5, pp. 417-419, 2004.

Bibliografía relacionada a la implementación

21. F. Minera, *XML La Guía Total del Programador*, Primera edición, MP Ediciones, Argentina, 2006.

22. G. Alonso, F. Casati, H. Kuro, V. Manchiraju, *Web Services*, 1st Edition, Springer, USA, 2003.

23. A. H. Scragg, *Biotechnology for engineers*, University of Sheffield, Dep. of Molecular Biology and Biotechnology, John Wiley and Sons, 1988.

24. J. D. Bulock, *Introduction to Basic Biotechnology* University of Manchester, Dep. of Chemistry, John Wiley and Sons, 1987.

25. D. C. Montgomery, *Diseño y análisis de experimentos*, 2da edición, editorial Limusa, 2003.

26. P. P. Mager, *Design statistics in Pharmacochemistry*, University of Leipzig, Germany, John Wiley and Sons, 1991.

27. J. N. Miller and J. C. Miller, *Estadística y Quimiometría para Química Analítica*, Cuarta Edición, Prentice Hall, España, 2002.

28. T.D.V.Swinscow and M.J.Campbell, *Statistics at Square One*, 10th Edition, BMJ Books, UK, 2006.

29. J. E. Freund, *Modern Elementary Statistics*, 4th Edition, Prentice Hall, USA, 1973.

30. G. Hartley, *Estadística Básica*, Primera Edición, Ediciones del Castillo, España 1973.

31. E. R. Harold, *Java I/O*, 2nd Edition, O'Reilly, USA, 2006.

32. P. Niemeyer and J. Knudsen, *Learning Java*, 2nd Edition, O'Reilly, USA, 2002.

33. H. Schildt and J. Holmes, *El Arte de Programar en Java*, Primera Edición, Mc Graw Hill, México, 2004.

34. H. M. Deitel and P. J. Deitel, *Java How to Program*, 3rd Edition, Prentice Hall, USA, 1999.

35. J. R. Hubbard, *Programming with Java*, 1st Edition, Mc Graw Hill, Malaysia, 1999.

36. J. R. Hubbard, *Data Structures with Java*, 1st Edition, Mc Graw Hill, USA, 2001.

37. N. Dale, C. Weems and M. Headington, *Introducction to Java and Software design*, 1st Edition, Jones and Bartlett, USA, 2001.

38. D. C. Marinescu, *Internet-Based Workflow Management: Toward a Semantic Web*, John Wiley and Son, 2002.

39. D. Giblin and R. Lam, *Programming Workflow Aplications with Domino*, R&D Books, 2000.

Consultas de Internet

40. http://sourceforge.net/projects/taverna–*Tutorial de Taverna*

41. http://taverna.sourceforge.net–*Taverna Project*

42. http://taverna.sourceforge.net–*Taverna User Manual 1.5*

43. http://ws.apache.org–*Web Services-Axis*

44. http://www.w3c.org–*World Wide Web Consortium*

45. http://w3schools.com–Tutorial

46. http://forum.java.sun.com–Consulta

47. http://java.sun.com/docs/book/tutorial/getstarted/problems/index.html–Tutorial

48. http://tomcat.apache.org/tomcat-5.5-doc/index.html–Consulta

49. http://www.osmosislatina.com/axis/webserviceswsdl.html–Consulta

50. http://www.omii.ac.uk/docs/2.3.0/reference/apache-axis/wsdd/wsdd.htm–Consulta

51. http://www.javaranch.com/newsletter/May2002/axis.html–Consulta

52. http://janas.objectweb.org/current/doc/howto/WebServices.html–Consulta

53. http://www.graphviz.org–Consulta

54. http://www.chuidiang.com/java/classpath/classpath.php–Tutorial

55. http://www.tech-recipes.com/java_programming_tips826.html–Consulta

56. http://www.203.2.177.22/tutorial/html/chap9.html–Tutorial

57. http://www.docjar.com/html/api/org/apache/axis/client/Service.java.html–Consulta

58. http://www-128.ibm.com/developerworks/library/ws-intwsdl–Tutorial

59. http://gdp.globus.org–Consulta

60. http://jamessmith73.googlepages.com/wsclient–Consulta

61. http://cheetah.cs.umb.edu/furums/archive/index.php–Consulta

62. http://javadeveloper.co.in/java-example/java-string-split-example.html–Consulta

63. http://webdia.cem.itesm.mx/ac/rogomez/Tutorial-LengC/binarios.html—Tutorial

64. http://mindprod.com/jgloss/binaryformats.html–Consulta

65. http://www.dbws.net/blog/?p=8–Consulta

66. http://www.electroduendes.com/blog/procesado-de-imagenes-en-java/–Consulta

67. http://www.freshsources.com/Apr01.html–Consulta

68. http://www.javaworld.com/javaworld/jw-05-2000/jw-0505-servlets.html?page=1–Tutorial

69. http://forum.java.sun.com/thread.jspa?threadID=5183630–Consulta

70. http://www.theserverside.com/discussions/thread.tss?thread_id=44653–Consulta

71. http://www.clubdevelopers.com–Consulta

72. http://leepoint.net/notes-java/GUI-lowlevel/graphics/43buffimage.html–Consulta

73. http://www.uky.edu/Provieders/datatypes.html–Consulta

74. http://catcode.com/pngencoder–Consulta

75. http://www.libpng.org/pub/png–Consulta

76. http://java.sun.com/j2se/1.4.2/docs/guide/imageio/index.html–Tutorial

77. http://www.trianacode.org

78. http://www.kepler-project.org

79. http://software.sci.utah.edu/scirun.html

80. http://tmitwww.tm.tue.nl/research/patterns

81. http://ptolemy.eeecs.berkeley.edu

82. http://research.cs.vt.edu/pse/intro.html

83. http://gforge.nci.nih.gov/docman/view.php/244/8269/taverna_tom_oinn.doc