Metaherramienta para la generación de aplicaciones científicas basadas en *workflows*

Rubén Salado-Cid, José Raúl Romero y Sebastián Ventura

Dpto. de Informática y Análisis Numérico
Universidad de Córdoba, Campus de Rabanales, 14071 Córdoba
{rsalado,jrromero,sventura}@uco.es

Resumen. El uso de la programación visual en el ámbito científico ha contribuido al desarrollo de aplicaciones que facilitan la realización de experimentos. Actualmente, existen aplicaciones para trabajar sobre un único dominio, limitadas a procedimientos propios de ese dominio, y aplicaciones multidominio, cuya complejidad para la configuración de sus elementos supone un gran esfuerzo para el usuario. Por tanto, es necesario ofrecer flexibilidad para trabajar sobre diversos dominios y permitir una adaptación intuitiva a dominios conocidos por el usuario. En este trabajo se presenta una metaherramienta para la generación automática de aplicaciones adaptadas a un dominio, o conjunto de dominios, para la composición y ejecución de *workflows* científicos en términos de procesos locales y servicios remotos. Estas nuevas aplicaciones disponen de una infraestructura que proporciona interoperabilidad con aplicaciones externas, presentan interfaces de usuario personalizadas y abstraen al usuario final de la complejidad de configuración al ofrecer elementos de trabajo ya adaptados a su campo.

Palabras clave: workflow científico, metaherramienta, programación visual, servicios remotos

1. Introducción

Actualmente, el uso de aplicaciones basadas en *workflows* está muy extendido, con un especial crecimiento en el ámbito científico en la última década. Este tipo de aplicaciones, gracias a la utilización del paradigma de la programación visual [14], permite a un usuario la definición y gestión de algoritmos sin la necesidad de ser experto en programación. Para su representación visual se dispone de una interfaz gráfica de usuario que, haciendo uso de diversos elementos visuales, proporciona acceso a una serie de procesos o recursos (de forma local o remota), aislando al usuario final de los aspectos de implementación internos, como el lenguaje de programación utilizado o la plataforma subyacente.

Un objetivo común de estos sistemas es el de facilitar a científicos un acceso transparente a numerosos procesos, herramientas y recursos útiles en un determinado dominio, o conjunto de dominios, para la realización de experimentos y el posterior análisis de los resultados obtenidos. La complejidad de los problemas

abordados en estos experimentos hace necesario que expertos de distintos ámbitos trabajen conjuntamente para alcanzar una solución final. Sin embargo, la diversidad de tecnologías utilizadas en el desarrollo de este tipo de aplicaciones, como el modelo de computación subyacente, el lenguaje de *workflow* utilizado o el entorno de ejecución, hace que lograr este objetivo no sea algo trivial [8].

Dada la gran diversidad de ámbitos científicos (bioquímica, genética, biomedicina, etc.), han surgido una serie de herramientas multidominio [6,12,15] que ofrecen distintos elementos configurables para acceder a recursos y procesos propios de cada dominio. A pesar de la versatilidad que presentan este tipo de aplicaciones, su complejidad y el tiempo de diseño aumenta con respecto a aplicaciones más específicas debido a la constante necesidad de configuración de parámetros, adecuación de procedimientos, etc. Así pues, la complejidad y la falta de adaptación del entorno de trabajo al dominio específico, hace que sea necesario buscar mecanismos que ofrezcan flexibilidad para trabajar sobre múltiples dominios, a la vez que permitan al usuario una adaptación y parametrización más intuitiva para los dominios que le resultan familiares.

En este artículo se presenta una metaherramienta para la creación de aplicaciones científicas basadas en *workflows*, orientada a dos perfiles diferenciados de usuario. Los usuarios con perfil avanzado son los encargados de personalizar y generar las nuevas aplicaciones específicas para cada dominio. Para ello, se ofrecen mecanismos que les permiten crear y adaptar una serie de elementos, como procesos, fuentes de datos, servicios remotos o visualizadores de datos, que formarán parte de las aplicaciones generadas. Por otra parte, los usuarios con perfil de experto en el dominio son los usuarios finales de dichas aplicaciones, que no necesariamente requieren conocer su estructura interna o un lenguaje de programación. Las aplicaciones generadas presentan una interfaz personalizada según los criterios del desarrollador, ofrecen todos los elementos funcionales básicos para la composición de *workflows*, permiten la ejecución de los *workflows* definidos tanto con la propia aplicación como con otras aplicaciones generadas con la metaherramienta, y proporcionan un acceso directo a los procesos, servicios y recursos adaptados al dominio, abstrayendo al usuario experto de la complejidad de su configuración. Además, se ha desarrollado un lenguaje específico de dominio (DSL, *Domain-Specific Language*) para la descripción de los *workflows*, cuya sintaxis concreta es interpretada por el motor de ejecución de la herramienta. Para lograr la integración con herramientas externas, su sintaxis abstracta, descrita en términos del metamodelo, permite obtener una versión compatible con otros lenguajes de *workflows* mediante transformaciones de modelos.

En el resto del artículo se presenta en profundidad la metaherramienta propuesta. En la Sección 2 se muestra el trabajo relacionado, incluyendo algunas de las herramientas más utilizadas en este ámbito. En la Sección 3 se presenta el DSL desarrollado para la ejecución de los *workflows*. En la Sección 4 se detalla la arquitectura general y los módulos de la metaherramienta. La Sección 5 expone un caso de estudio realizado para la generación de una aplicación sobre un dominio específico. Finalmente, en la Sección 6 se presentan las conclusiones y se explican algunas líneas de trabajo futuro.

2. Trabajo relacionado

Los sistemas basados en *workflows* han sido utilizados tradicionalmente para la automatización de los procesos de negocio [2]. Los lenguajes utilizados en este ámbito [11] usan flujos de control para especificar el orden de realización de las tareas definidas dentro del propio *workflow*. Esta tecnología ha sido adoptada por la comunidad científica para la automatización de los experimentos [1], si bien presentan diferencias notables con los primeros [18]. Debido al uso intensivo de los datos que se lleva a cabo durante un experimento, los lenguajes utilizados para la definición de *workflows* científicos utilizan flujos de datos para definir el orden de ejecución de sus elementos [19].

Dentro del ámbito científico, se encuentran disponibles numerosas aplicaciones basadas en *workflows* que permiten trabajar sobre dominios específicos, como minería de datos [3,7], bioinformática [16], neurociencia [13], etc. También se han desarrollado herramientas independientes del dominio, como Taverna [15], Kepler [12] o Triana [6]. Taverna permite la definición y ejecución de *workflows* científicos proporcionando acceso a recursos locales y remotos, así como a herramientas de análisis. Se integra con otras herramientas como myExperiment [17], para el intercambio de *workflows*, y BioCatalogue [4], para el descubrimiento de servicios web para el ámbito científico. Kepler es una herramienta diseñada para ayudar a científicos, analistas y programadores a desarrollar, ejecutar e intercambiar modelos y análisis en numerosos ámbitos científicos, cuya principal característica es el modelado orientado a actores [5]. Triana es un entorno para la resolución de problemas, que proporciona herramientas para el análisis de datos y cuenta con un gran número de componentes para trabajar en dominios como el procesamiento de audio, texto, etc. La principal limitación de estas herramientas radica en la complejidad para ser adaptadas a un dominio específico, y en la imposibilidad de generar aplicaciones particularizadas con elementos de trabajo ya configurados. Por ello, resultaría de interés contar con una metaherramienta en el campo de los *workflows* científicos que facilitara esta adaptación.

De hecho, ya se han realizado esfuerzos para facilitar el metadesarrollo de aplicaciones en otros ámbitos. En el contexto de MDE (*Model Driven Engineering*), por ejemplo, existen herramientas como GMF [10], para la generación de editores gráficos a partir de la definición del modelo de dominio que determina sus características, o Xtext [9], para el desarrollo de IDEs (*Integrated Development Environment*) de lenguajes específicos del dominio, cuya infraestructura incluye analizadores, enlazadores o intérpretes del lenguaje. Hasta donde sabemos, no existen metaherramientas que permitan la generación automática de aplicaciones de gestión de *workflows* científicos adaptadas a dominios específicos.

3. Lenguaje específico para la ejecución de *workflows*

La descripción a alto nivel de los procesos, servicios y recursos definidos en un *workflow* debe ser convertida en una versión ejecutable por la plataforma. Para especificar de forma precisa toda la información relevante para la ejecución

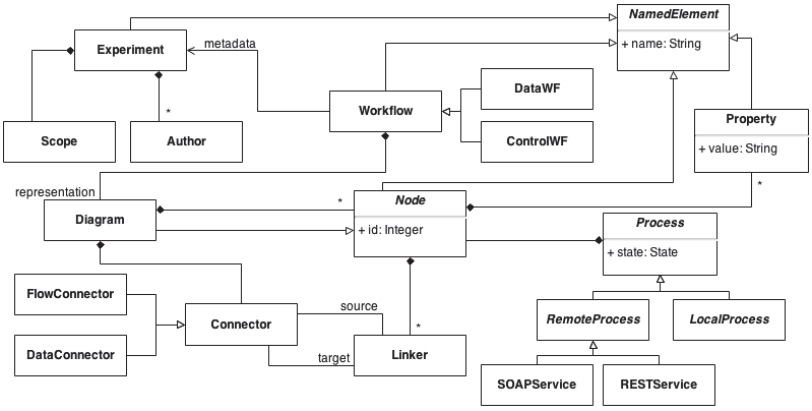

Fig. 1. Vista parcial simplificada de la sintaxis abstracta del lenguaje.

de *workflows*, se ha modelado un DSL propio, cuya sintaxis abstracta ha sido definida en términos de meta-modelos. Por motivos de espacio, la Figura 1 muestra parcialmente el meta-modelo que define el lenguaje, el cual está dividido en diferentes paquetes, según la especialización y funcionalidad de sus elementos: *workflow*, entrada y salida de datos, procesos y estructuras de control.

Workflow. Define la metainformación del experimento (*Experiment*) y del *workflow* a ejecutar. Soporta dos tipos, dependiendo de si es dirigido por flujos de control (*ControlWF*) o por flujos de datos (*DataWF*). Su representación es definida mediante un diagrama (*Diagram*), que cuenta con una serie de nodos (*Node*) y conectores (*Connector*).

Entrada y salida de datos. Incluye los elementos que definen las fuentes de datos heterogéneas, locales y remotas, de entrada y salida. Cada elemento está compuesto por un único enlazador (*Linker*), que representa un punto de interacción en una conexión de flujo de datos (mediante *DataConnector*) para la incorporación, visualización o almacenamiento de datos.

Proceso. Incluye la declaración de todo procedimiento, local o remoto, que manipula unos datos de entrada para generar datos de salida. Un procedimiento local (*LocalProcess*) es aquel que se ejecuta en la propia máquina donde se encuentra la aplicación. Puede ser de tres tipos, según su alcance y funcionalidad: proceso complejo (transforma un conjunto de entradas en un conjunto de salidas mediante un procedimiento complejo), diagrama (*workflow* que define un procedimiento anidado) y filtro (únicamente dedicado a la manipulación sencilla de los flujos de datos como, por ejemplo, realizar conversiones de tipos). Un procedimiento remoto (*RemoteProcess*) define un servicio externo, al que actualmente se puede acceder mediante REST o SOAP/WSDL. Cada elemento está compuesto por tantos enlazadores como datos de entrada y de salida defina el procedimiento, y permite tanto conexiones de flujo de control (*FlowConnector*) como de datos.

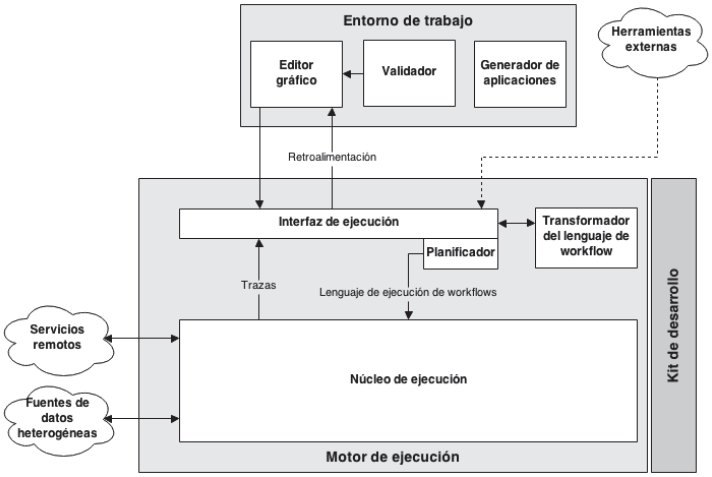

Fig. 2. Arquitectura general de la metaherramienta.

Estructuras de control. Incluye los elementos que determinan el flujo de ejecución. Se consideran dos tipos de estructuras de control: esquemas condicionales y bucles. Los esquemas condicionales definen las salidas activas tras la evaluación de una condición. Pueden ser de tipo *si-entonces* (*if-else*) y de tipo *en-caso-de* (*switch*). Un bucle define el valor de una variable que, junto al uso de esquemas condicionales, se utiliza para la declaración de iteraciones.

4. Arquitectura general de la herramienta

La Figura 2 muestra el esquema de la arquitectura general de la metaherramienta. Es una arquitectura modular que separa sus componentes en dos elementos principales: el entorno de trabajo (que permite al usuario la composición de procesos y definición de *workflows*, además de la configuración, personalización y generación de las nuevas aplicaciones) y el motor de ejecución (que procesa los *workflows* compatibles con la plataforma). A este módulo se accede además mediante un kit para desarrolladores que permite extender su funcionalidad e interoperabilidad con herramientas y lenguajes externos.

4.1. Entorno de trabajo

El entorno de trabajo, que ha sido desarrollado como *plugin standalone* de la plataforma Eclipse, posee una interfaz gráfica de usuario que consta a su vez de distintos componentes. Para la creación de *workflows* se ofrece un editor gráfico, que dispone de los elementos necesarios para su composición, definición y control de ejecución. Para el desarrollo y creación automática de las nuevas aplicaciones se ofrecen un conjunto de *wizards* y mecanismos para la selección de recursos, procesos y servicios que formarán parte de las mismas.

Editor gráfico. Está compuesto por una paleta de herramientas y un área de composición de diagramas, orientada al experto en el dominio, para la creación visual de *workflows* complejos y jerárquicos. A través de esta paleta es posible seleccionar cada uno de los nodos y conectores, tal y como se definen en el DSL, que conformarán el *workflow*. El editor también permite la configuración de cada elemento y ofrece información precisa sobre su ejecución y depuración.

Validador. Es el módulo encargado de validar en vivo el *workflow* durante la fase de creación del mismo. Comprueba que se cumplen las restricciones impuestas por la notación abstracta del lenguaje y notifica al usuario sobre posibles errores cometidos en la composición y configuración de los elementos que lo conforman.

Generador de aplicaciones. Permite la generación automática de aplicaciones específicas, particularizadas con elementos de trabajo previamente adaptados al dominio. Para ello, se debe introducir la información relativa a estos elementos, junto con la información necesaria para personalizar su interfaz gráfica. Estas nuevas aplicaciones, exportadas como *plugin standalone* de Eclipse, proporcionan un entorno para la composición y ejecución de *workflows*, con acceso a procedimientos ya adaptados al dominio, tanto visual como funcionalmente.

4.2. Motor de ejecución

Se ha desarrollado un motor de ejecución propio que se encarga de invocar servicios y recursos remotos, ejecutar procedimientos locales, interpretar datos de entrada y salida, así como de calcular el flujo de ejecución y de emitir trazas. Además, permite que la ejecución de los *workflows* pueda ser realizada tanto de forma regular como en modo depuración.

Interfaz de ejecución. El motor de ejecución gestiona la comunicación con módulos externos, ofreciendo flexibilidad y permitiendo así la escalabilidad de la herramienta. Esta interfaz es la encargada de recibir las peticiones de ejecución de *workflows* y de proporcionar la retroalimentación requerida.

Transformador de lenguajes. A partir de la sintaxis abstracta del DSL, permite realizar transformaciones entre lenguajes de *workflows*. Así, permite la interoperabilidad con aplicaciones externas basadas en *workflows* al permitir ejecutar otras notaciones concretas desde el módulo de ejecución, independientemente de la interfaz de usuario utilizada.

Planificador. Realiza un procesamiento previo sobre la representación a alto nivel de los procedimientos y recursos definidos en el *workflow* para obtener una versión ejecutable del mismo. Este procesamiento trata de optimizar la ejecución teniendo en cuenta distintos aspectos, como el tipo de *workflow* definido (dirigido por flujos de control o por flujo de datos), la cantidad de memoria disponible, las unidades o nodos de ejecución de la plataforma, la tipología de dicha plataforma de ejecución, etc.

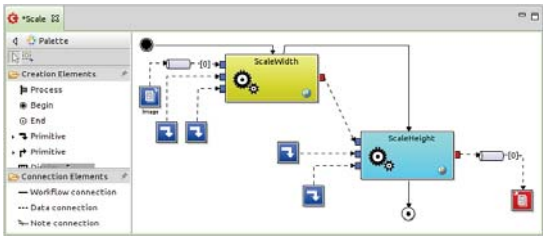

Fig. 3. Flujo de control sencillo invocando a servicios remotos.

Núcleo de ejecución. A partir de una descripción ejecutable de un *workflow* obtenido por el planificador, este componente ejecuta cada uno de sus elementos según las pautas establecidas. Durante este proceso, se generará la información sobre la ejecución en forma de trazas, junto con los resultados del experimento.

Kit de desarrollo. Para facilitar la extensibilidad tanto del motor de ejecución como de la herramienta, se han desarrollado una serie de interfaces de programación (actualmente en Java), que permiten a un desarrollador incrementar la variedad de tipos de datos con los que puede trabajar la herramienta, ampliar múltiples funcionalidades propias del sistema, así como extender el catálogo disponible de filtros, procesos, visualizadores de datos, etc.

5. Generación de una herramienta para el procesamiento de imágenes: caso de estudio

Para mostrar el funcionamiento de la metaherramienta a la hora de generar nuevas aplicaciones específicas a un dominio, se trabajará sobre un caso de estudio focalizado en la manipulación de imágenes. Esta nueva aplicación generada contará con una serie de procedimientos, desarrollados mediante programación visual, abstrayendo al usuario final de toda la complejidad de diseño y configuración, y que le permitirán realizar tratamientos específicos sobre imágenes de entrada, invocando servicios remotos para realizar su procesamiento.

En primer lugar, se deberán crear (o bien buscar ya existentes) y configurar los procesos que formarán parte de la nueva aplicación, mediante el editor gráfico. En la Figura 3 se muestra la paleta con los elementos del lenguaje disponibles, así como un *workflow* ya definido para modificar el tamaño de una imagen de entrada, que contiene dos procesos remotos que invocan a un servicio web, cuyas entradas y salidas son descubiertas automáticamente por la herramienta y dispuestas en el diagrama (analizando el WSDL en el caso de los servicios SOAP, y a partir de la URL en el caso de los servicios REST), y dos filtros para adaptar el tipo de dato al requerido por el servicio y por el elemento de salida, respectivamente. El resto de elementos se corresponden con las imágenes de entrada y salida, con cuatro datos de entrada de usuario referidos a las coordenadas de escalado de la imagen, y con los puntos de inicio y fin del *workflow*. Durante esta

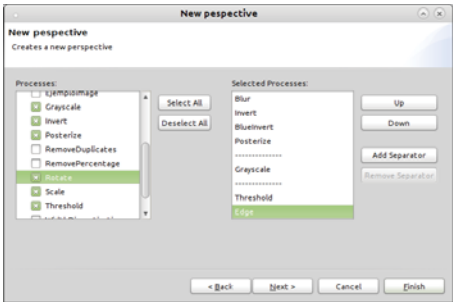

Fig. 4. Configuración y selección de procesos (ventana de diálogo).

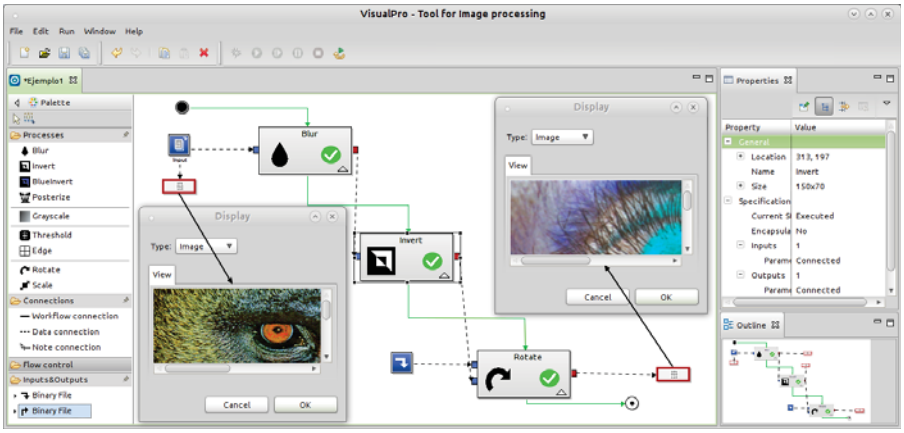

Fig. 5. Nueva herramienta generada para la manipulación de imágenes.

fase, el usuario dispone del kit de desarrollo para extender la funcionalidad de la metaherramienta incorporando nuevos procesos locales, si fuera necesario.

Una vez creados todos los *workflows* correspondientes a los procedimientos del dominio, se configurará la nueva aplicación a través del componente generador de herramientas. Para ello, el usuario deberá introducir información relativa a la interfaz gráfica, como el nombre de la nueva aplicación, una breve descripción de la misma y otra metainformación. Igualmente, se deberá proveer información sobre los procedimientos creados y configurados anteriormente, ya adaptados al dominio (Figura 4). Tras indicar estos elementos, y una vez personalizada y parametrizada la interfaz, se procede a la generación automática de la aplicación. Como se observa en la Figura 5, el entorno generado y los procesos disponibles, así como la descripción de los elementos y el tipo de *workflow*, se ha personalizado al dominio concreto, ocultando los detalles de implementación al usuario final. De esta manera, se ofrece una aplicación más intuitiva en su uso y adaptable al tipo de experimento objetivo.

6. Conclusiones y trabajo futuro

En este trabajo se ha descrito la primera metaherramienta para la generación de aplicaciones basadas en *workflows* en el ámbito científico. Las herramientas más genéricas existentes no abstraen al usuario final de la complejidad existente a la hora de extender su funcionalidad, o de configurar y adaptarla a un dominio concreto. Por tanto, con esta metaherramienta y su orientación a distintos perfiles de usuario se consigue mitigar este problema de cara al usuario final, proporcionando los mecanismos para crear, configurar y generar nuevas aplicaciones adaptadas al dominio. Para la definición y ejecución de los *workflows* se ha modelado un DSL cuya sintaxis abstracta permite la transformación desde y hacia otros lenguajes de *workflows*, permitiendo la interoperabilidad con otras herramientas. La arquitectura de la plataforma da soporte a estas funcionalidades presentando una estructura modular y extensible, ofreciendo además un kit de desarrollo para que programadores externos puedan ampliar y adaptar múltiples funcionalidades del sistema. Finalmente, para ilustrar el funcionamiento de esta herramienta, se ha mostrado un caso de estudio sencillo, en el que se ha generado una aplicación basada en *workflows* para un dominio específico. La aplicación resultante presenta una interfaz de usuario personalizada por el desarrollador, junto con los recursos y procesos basados en servicios seleccionados previamente que permiten al usuario experto trabajar sobre su propio dominio.

La línea a seguir en el trabajo futuro de la herramienta consiste en dotarla de un mayor número de mecanismos para la personalización de las aplicaciones generadas. Se desarrollará un catálogo de transformaciones de modelos para ofrecer compatibilidad con otros lenguajes de *workflows*. También se trabajará en la integración del motor de ejecución con otras plataformas tecnológicas (p. ej. computación distribuida, *grid computing*, etc.) para aumentar su rendimiento.

Agradecimientos

Trabajo apoyado por el Ministerio de Ciencia y Tecnología, proyecto TIN2011-22408, y fondos FEDER.

Referencias

1. Akram, A., Meredith, D., Allan, R.: Evaluation of BPEL to scientific workflows. In: Cluster Computing and the Grid, 2006. CCGRID 06. Sixth IEEE International Symposium on. vol. 1, pp. 269–274 (May 2006)
2. Barker, A., Hemert, J.: Scientific workflow: A survey and research directions. In: Wyrzykowski, R., Dongarra, J., Karczewski, K., Wasniewski, J. (eds.) Parallel Processing and Applied Mathematics, Lecture Notes in Computer Science, vol. 4967, pp. 746–753. Springer Berlin Heidelberg (2008)
3. Berthold, M., Cebron, N., Dill, F., Gabriel, T., Kötter, T., Meinl, T., Ohl, P., Sieb, C., Thiel, K., Wiswedel, B.: KNIME: The Konstanz Information Miner. In: Data Analysis, Machine Learning and Applications, pp. 319–326. Studies in Classification, Data Analysis, and Knowledge Organization, Springer (2008)

4. Bhagat, J., Tanoh, F., Nzuobontane, E., Laurent, T., Orlowski, J., Roos, M., Wolstencroft, K., Aleksejevs, S., Stevens, R., Pettifer, S., Lopez, R., Goble, C.A.: BioCatalogue: a universal catalogue of web services for the life sciences. Nucleic Acids Research 38(Web-Server-Issue), 689–694 (2010)
5. Bowers, S., Ludäscher, B.: Actor-oriented design of scientific workflows. In: Proc. of the International Conference on Conceptual Modeling (ER). pp. 369–384 (2005)
6. Churches, D., Gombás, G., Harrison, A., Maassen, J., Robinson, C., Shields, M.S., Taylor, I.J., Wang, I.: Programming scientific and distributed workflow with Triana services. Concurrency and Computation: Practice and Experience 18(10), 1021–1037 (2006)
7. Demšar, J., Zupan, B., Leban, G., Curk, T.: Orange: From experimental machine learning to interactive data mining. In: Boulicaut, J.F., Esposito, F., Giannotti, F., Pedreschi, D. (eds.) Knowledge Discovery in Databases: PKDD 2004, Lecture Notes in Computer Science, vol. 3202, pp. 537–539. Springer (2004)
8. Elmroth, E., Hernández, F., Tordsson, J.: Three fundamental dimensions of scientific workflow interoperability: Model of computation, language, and execution environment. Future Generation Comp. Syst. 26(2), 245–256 (2010)
9. Eysholdt, M., Behrens, H.: Xtext: Implement your language faster than the quick and dirty way. In: Proceedings of the ACM International Conference Companion on Object Oriented Programming Systems Languages and Applications Companion. pp. 307–309. SPLASH '10, ACM (2010)
10. Gronback, R.C.: Eclipse Modeling Project: A Domain-Specific Language (DSL) Toolkit. Addison-Wesley Professional, 1 edn. (2009)
11. Hepp, M., Hinkelmann, K., Karagiannis, D., Klein, R., Stojanovic, N. (eds.): Proceedings of the Workshop on Semantic Business Process and Product Lifecycle Management SBPM 2007, Innsbruck, Austria, June 7, 2007, CEUR Workshop Proceedings, vol. 251. CEUR-WS.org (2007)
12. Ludäscher, B., Altintas, I., Berkley, C., Higgins, D., Jaeger, E., Jones, M., Lee, E.A., Tao, J., Zhao, Y.: Scientific workflow management and the Kepler system. Concurrency and Computation: Practice and Experience 18(10), 1039–1065 (2006)
13. MacKenzie-Graham, A., Payan, A., Dinov, I., Horn, J., Toga, A.: Neuroimaging data provenance using the LONI Pipeline workflow environment. In: Freire, J., Koop, D., Moreau, L. (eds.) Provenance and Annotation of Data and Processes, Lecture Notes in Computer Science, vol. 5272, pp. 208–220. Springer (2008)
14. Myers, B.: Taxonomies of visual programming and program visualization. Journal of Visual Languages and Computing 1(1), 97–123 (1990), cited By (since 1996)78
15. Oinn, T., Greenwood, M., Addis, M., Ferris, J., Glover, K., Goble, C., Hull, D., Marvin, D., Li, P., Lord, P.: Taverna: Lessons in creating a workflow environment for the life sciences. Concurrency and Computation: Practice and Experience 18(10), 1067–1100 (2006)
16. Rampp, M., Soddemann, T., Lederer, H.: The MIGenAS integrated bioinformatics toolkit for web-based sequence analysis. Nucleic Acids Research 34(Web-Server-Issue), 15–19 (2006)
17. Roure, D.D., Goble, C., Bhagat, J., Cruickshank, D., Goderis, A., Michaelides, D., Newman, D.: myExperiment: Defining the social virtual research environment. In: 4th IEEE International Conference on e-Science. pp. 182–189. IEEE Press (2008)
18. Yildiz, U., Guabtni, A., Ngu, A.: Business versus scientific workflows: A comparative study. In: Services - I, 2009 World Conference on. pp. 340–343 (July 2009)
19. Yu, J., Buyya, R.: A taxonomy of workflow management systems for grid computing. Journal of Grid Computing 3(3-4), 171–200 (2005)