Cumbia – Modelos Ejecutables para construir Aplicaciones basadas en Workflows

Mario Sánchez[1 2], Jorge Villalobos[1], and Dirk Deridder[2]

[1] Grupo de Construcción de Software (CSW)
Universidad de los Andes, Bogotá, Colombia
{mar-san1,jvillalo}@uniandes.edu.co
[2] Software Languages Lab (SOFT)
Vrije Universiteit Brussel, Bruselas, Bélgica
Dirk.Deridder@vub.ac.be

Abstract. Este artículo resume una parte del trabajo realizado dentro del proyecto de investigación Cumbia. Este resumen incluye una breve introducción al contexto de las aplicaciones basadas en workflows, la exposición del problema principal que resolvemos y de la estrategia de solución (la plataforma Cumbia), la presentación de algunos de los resultados obtenidos, y una breve comparación con trabajos relacionados. El artículo incluye referencias a las publicaciones que presentan en detalle la estrategia de solución y sus resultados.

1 Aplicaciones basadas en Workflows

El proyecto Cumbia está orientado a solucionar problemas que se presentan en el contexto de las aplicaciones basadas en workflows. Según la definición de [1], un *workflow* es una colección de tareas organizadas siguiendo la estructura de un proceso de negocio. Estas tareas pueden ser: tareas enteramente manuales realizadas por humanos (por ejemplo verificar la existencia de un domicilio); tareas automatizadas realizadas por aplicaciones (por ejemplo generar automáticamente una factura), o tareas en las cuales los humanos usan sistemas informáticos (por ejemplo para llenar un formulario en línea). Los *Workflow Management Systems* (WfMS) son sistemas que sirven para automatizar la ejecución de workflows, es decir que sirven para definir, ejecutar, controlar y monitorear la ejecución de las tareas, siguiendo el orden establecido. A su vez, éstas interactúan con aplicaciones externas y permiten la participación de usuarios [1][2].

Los WfMS usan además un lenguaje para la definición de los workflows (WfDL), y cada WfDL tiene características particulares que lo hacen adecuado para resolver cierto tipo de problemas, en cierto tipo de contexto. Los WfMS tienen que ofrecer entonces las funcionalidades necesarias para ejecutar workflows descritos usando el WfDL correspondiente.

2 El problema de soportar y adaptar WfDL

La variedad en los usos que se le da a los workflows [3][4] es la causa de la aparición de un gran número de lenguajes, entre los cuales hay algunos que son más comúnmente utilizados, como BPEL, BPMN, o XPDL. Sin embargo, además de estos lenguajes que son grandes, poderosos y relativamente independientes del contexto, también se han desarrollado cientos de lenguajes que se usan en contextos muy particulares [5], por ejemplo hospitales, aplicaciones científicas e e-learning, entre otros. El problema es que implementar motores para estos lenguajes es generalmente muy costoso, porque requiere que sean construidos prácticamente desde cero. Esto no solamente aplica para nuevos lenguajes, sino también para lenguajes existentes que deban adaptarse a nuevas funcionalidades.

Hay varias causas para estos problemas. Por un lado, los WfDL y los WfMS existentes están muy fuertemente acoplados, haciendo muy difícil adaptar las aplicaciones existentes a un nuevo lenguaje. Además estas herramientas generalmente tienen capacidades de extensión y adaptación muy limitadas. Por otro lado muchos lenguajes tienden a ser muy grandes para poder cubrir en amplio espectro de problemas. Además, muchos son también monolíticos y por lo tanto no se prestan para ser modularizados con facilidad debido al alto acoplamiento entre sus elementos. Como consecuencia, la implementación de motores para estos lenguajes es más costosa de lo que sería si los lenguajes fueran pequeños y altamente especializados, y además las posibilidades de reutilización son muy bajas.

El objetivo de esta investigación es entonces proponer una aproximación que permita la construcción a bajo costo de WfMS para nuevos lenguajes, y que también facilite su posterior adaptación y extensión.

La solución que proponemos está basada en las siguientes hipótesis:

1. Es más fácil implementar y mantener lenguajes pequeños y especializados que lenguajes grandes y generales [6].
2. Es más barato diseñar e implementar un nuevo lenguaje pequeño y especializado, que intentar adaptar o extender un lenguaje muy grande y complejo [8][7].
3. Es posible modularizar la descripción de un workflow para que diferentes aspectos se definan usando diferentes lenguajes [9][10].
4. Si estos lenguajes son pequeños y especializados, es posible utilizar diferentes combinaciones de estos lenguajes dependiendo del contexto y del problema que se esté solucionando.
5. Si los workflows se modularizan los motores para cada lenguaje deben compartir características que los hagan *compatibles* y les permita interactuar.
6. Es posible construir un núcleo que ofrezca funcionalidades básicas y sobre el cual puedan construirse los motores requeridos a un costo bajo.

3 Elementos de la solución: la plataforma Cumbia

La solución propuesta al problema planteado se basa en 3 ideas principales, que son implementadas en la plataforma Cumbia.

Descomposición basada en Dimensiones La primera idea es una estrategia para modularizar las definiciones de workflows y permitir la integración de varios lenguajes. En nuestra propuesta, esta división se realiza por dimensiones y cada dimensión se describe usando un lenguaje especializado [9].

En las aplicaciones basadas en workflows, la dimensión más importante es la de control, la cual especifica qué tareas hay que realizar, en qué orden y bajo qué condiciones. Otras dimensiones muy comunes son la de recursos, que describe los recursos utilizados o consumidos durante la ejecución de un proceso, la dimensión de tiempo, que describe características de los procesos desde el punto de vista del tiempo, y la dimensión de datos.

Cualquier división en dimensiones es arbitraria, y es posible identificar diferentes dimensiones según el contexto. Sin embargo, algunas divisiones son más útiles que otras porque desacoplan los elementos para que sean más fáciles de reemplazar o reutilizar. Por ejemplo, los elementos que aparecen en la dimensión de control varían mucho entre contextos. En cambio, las restricciones de tiempo tienen estructuras y características bastante similares en todos los contextos, así que se pueden reutilizar con más facilidad. Para concluir, lo más importante al identificar dimensiones es que no haya aspectos repetidos, y que se puedan relacionar claramente los elementos de diferentes dimensiones: por ejemplo las restricciones de tiempo deben poder relacionarse con elementos de la dimensión de control como actividades y procesos.

Plataforma de ejecución basada en Modelos Ejecutables A partir de la estrategia de separación en dimensiones, surge la necesidad de una plataforma para representar y ejecutar cada una de esas dimensiones. Además, esta plataforma tiene que ser muy expresiva para poder soportar todas las dimensiones que puedan aparecer en cualquier contexto. Para lograr esto diseñamos la plataforma Cumbia, la cual se basa en modelos ejecutables extensibles que se construyen con especializaciones de un elemento básico de coordinación llamado *objeto abierto*. Los objetos abiertos se basan en máquinas de estado sincronizadas por medio de eventos y ofrecen características que los hacen adecuados para representar cualquier dimensión de un workflow [11, 12].

Para utilizar objetos abiertos primero es necesario definir *metamodelos* para cada una de las dimensiones. Usando estos metamodelos es posible describir *modelos*, los cuales representan las partes de un workflow. Algo importante es que estos metamodelos pueden variar, es decir que pueden sufrir modificaciones para acomodar nuevos requerimientos, o pueden incluso ser reemplazados en su totalidad [13]. Sin embargo, es importante también tener en cuenta que esto puede traer problemas de consistencia en la evolución [14].

Finalmente, la idea de usar la misma plataforma para todas las dimensiones tiene la ventaja adicional de permitir que otras herramientas (por ejemplo herramientas de monitoreo) se reutilicen en contextos diferentes [15].

Composición y coordinación de modelos ejecutables El último aspecto de nuestra solución es un mecanismo para componer y coordinar la ejecución

de los modelos ejecutables. Para esto también se aprovechan los mecanismos de coordinación entre objetos abiertos, y se utiliza un lenguaje llamado CCL (Cumbia Composition Language): CCL describe el tejido entre instancias de modelos ejecutables basados en Cumbia, utilizando suscripciones a eventos y *callbacks*. Además, como CCL depende únicamente de los objetos abiertos, puede utilizarse con cualquier metamodelo.

4 Algunos resultados

A continuación se presentan algunos de los experimentos realizados con Cumbia, y los principales resultados obtenidos.

Construcción de la plataforma. En términos prácticos, el principal resultado obtenido hasta el momento ha sido la construcción de la plataforma Cumbia, la cual implementa la estrategia presentada. El principal elemento de esta plataforma es el *Cumbia Kernel*, el cual ofrece el soporte para la ejecución de los objetos abiertos y que se utiliza como base para la construcción de motores para WfDL específicos. Además de este kernel, también hacen parte de la plataforma Cumbia herramientas tales como un editor para los metamodelos, y un framework para la construcción de pruebas automatizadas.

XPM y XTM. XPM (eXtensible Process Metamodel) fue el primer WfDL construido para Cumbia y es un lenguaje que se enfoca únicamente en la dimensión de control. XTM (eXtensible Time Metamodel) es un lenguaje para representar la dimensión de tiempo. Juntos, XPM y XTM sirven para definir workflows en los que las actividades y procesos tienen que cumplir con un conjunto de restricciones que dependen del tiempo.

Redes de Petri. Las redes de Petri son un formalismo para modelar sistemas concurrentes. Con mucha frecuencia son usadas para representar workflows y varios WfMS las usan para controlar la ejecución de los procesos. Como parte de nuestra investigación, también construimos una aplicación para ejecutar Redes de Petri utilizando Cumbia, la cual nos permitió evaluar la expresividad de los mecanismos de coordinación de los objetos abiertos.

YAWL. YAWL es un WfDL definido con el objetivo de soportar todos los patrones de control [16][17] y es un referente dentro del campo de la investigación sobre workflows. Por este motivo en este momento nos encontramos diseñando e implementando un metamodelo para YAWL, el cual nos permitirá compararnos con más facilidad con otras herramientas existentes.

Otros experimentos. La plataforma Cumbia también ha sido usada para en el desarrollo de Caffeine y de Alegre, que son aplicaciones para soportar los WfDL más utilizados (BPEL y BPMN, respectivamente). Otros experimentos muy interesantes han sido el desarrollo de Garabato, que se enmarca en el contexto de e-learning y permite la ejecución de especificaciones IMS-LD, y el desarrollo de PaperXpress, que es una aplicación para trabajo colaborativo basada en workflows. Finalmente, otro experimento realizado fue la integración de Cumbia con Eleggua [18], con la cual se logró utilizar el motor de XPM para orquestar la interacción de aplicaciones usando un mecanismo basado en aspectos y reglas ECA [19].

5 Trabajos relacionados

Los siguientes proyectos y herramientas atacan problemas similares a los que ataca Cumbia, o utilizan estrategias que de alguna forma han inspirado el diseño de la plataforma. Por motivos de espacio, esta presentación tiene que ser muy breve, pero se proporcionan referencias a la literatura más relevante.

Teniendo en cuenta que entre los objetivos de este trabajo está soportar con facilidad nuevos lenguajes, podría argumentarse que no es necesario diseñar y soportar nuevos lenguajes porque los actuales son comparativamente muy poderosos y expresivos. Sin embargo, la realidad es que hay cientos de WfDL y WfMS que han sido construidos porque los lenguajes más ampliamente difundidos se quedan cortos cuando tienen que resolver problemas en contextos muy especializados [5]. Por ejemplo, el caso de BPEL4PEOPLE [20] muestra la alta complejidad de una extensión especializada para BPEL, que aprovecha algunos de los puntos de extensión predeterminados del lenguaje. Otro ejemplo es Sedna, que aprovecha algunas ventajas ofrecidas por BPEL, pero que requirió la definición de un nuevo lenguaje y la construcción de nuevas herramientas [21].

AO4BPEL y Padus han también implementado estrategias para descomponer workflows [22][23]. Sin embargo, estas herramientas no modularizan dimensiones, sino que separan básicamente aspectos funcionales. A pesar de que esto ofrece ventajas, también tiene una limitante muy importante: los diferentes aspectos de un workflow se describen usando siempre el mismo lenguaje (BPEL). Otro proyecto similar es AMFIBIA [10], en el cual los worfklows se descomponen en *perspectivas*.

La idea de construir kernels reutilizables que permitan construir o extender con facilidad nuevos motores para workflows ha sido explorada desde varios puntos de vista en trabajos anteriores como Micro-Workflow citemicroWorkflow, MENTOR [24], Opera [25] y otros [26]. Sin embargo, ninguna de estas soluciones utiliza una descomposición basada en dimensiones, ni utiliza modelos ejecutables. Además, varios de ellos están limitados a modelos de ejecución particulares, y no necesariamente pueden adaptarse a cualquier nuevo lenguaje.

Una línea de investigación relacionada es la que intenta construir lenguajes de workflow muy expresivos, que puedan utilizarse como lenguajes ejecutables intermedios a los que se traduzcan procesos descritos con lenguajes especial-

izados de más alto nivel [16][27]. En Cumbia también se han realizado algunos experimentos en esa dirección, utilizando XPM como lenguaje intermedio. Sin embargo, esta aproximación tiene varios problemas, entre los cuales el más importante es recuperar información de alto nivel sobre la ejecución.

Finalmente, el trabajo realizado en esta investigación está estrechamente relacionado con la ingeniería guiada por modelos (MDE) y con el *multi-modelaje* (multi-modeling). En particular, la estrategia de Cumbia tiene similitudes con ambientes de metamodelado como GME [28], pero hay varias diferencias importantes en todo lo que tiene que ver con la estrategia de ejecución. Por ejemplo, Cumbia no usa código generado. Además, la idea de utilizar lenguajes específicos para cada dimensión, puede relacionarse fácilmente con lo que se hace en ambientes de multi-modelado como Ptolemy [29].

6 Los proyectos Cumbia y Caramelos

El trabajo de investigación presentado en este artículo ha sido realizado dentro del Grupo de Construcción de Software (TICsw)[3], el cual depende del Departamento de Ingeniería de Sistemas de la Universidad de los Andes. El proyecto Cumbia[4] inició en el año 2004, y desde entonces han participado en él cerca de 20 estudiantes de maestría dirigidos por Jorge Villalobos. En general, las ideas y resultados mencionados en este artículo son el resultado del trabajo conjunto del equipo de Cumbia. Sin embargo, cada tesis de maestría ha profundizado en algún tema particular relacionado con el desarrollo de la plataforma, o ha realizado validaciones en contextos específicos.

La participación de Mario Sánchez en el proyecto Cumbia inició como estudiante de maestría en el año 2005, y luego continuó como estudiante doctoral. Su trabajo ha estado enfocado en la construcción del núcleo de Cumbia (incluyendo la definición y soporte de los modelos ejecutables, la definición de CCL y la coordinación de la ejecución) y en la conceptualización de las ideas que respaldan la propuesta. Él también ha participado en la validación de la plataforma, incluyendo la construcción de XPM, y en la construcción de un framework para probar modelos basados en Cumbia.

El trabajo de Mario Sánchez como estudiante doctoral ha estado enmarcado por un acuerdo de cotutela entre la Universidad de los Andes y la Vrije Universiteit Brussel. Este acuerdo establece que hay dos co-directores de tesis (Jorge Villalobos y Dirk Deridder) y tiene su origen en el proyecto Caramelos[5]. Durante esta fase como estudiante doctoral, su trabajo ha sido patrocinado por Colciencias[6] y por el instituto VLIR del gobierno de Bélgica[7], a través del proyecto Caramelos.

[3] URL TICsw: http://ticsw.uniandes.edu.co/
[4] URL Cumbia: http://cumbia.uniandes.edu.co/
[5] URL Caramelos: https://ssel.vub.ac.be/caramelos/
[6] URL Colciencias: http://www.colciencias.gov.co/
[7] URL VLIR: http://www.vlir.be/

Referencias

1. Georgakopoulos, D., Hornick, M.F., Sheth, A.P.: An overview of workflow management: From process modeling to workflow automation infrastructure. Distributed and Parallel Databases **3**(2) (1995) 119–153
2. Alonso, G., Agrawal, D., Abbadi, E.A., Mohan, C.: Functionality and limitations of current workflow management systems. IEEE Expert **12**(5) (1997) 68–74
3. Palmer, N.: A survey of business process initiatives. url: http://wfmc.org/researchreports/Survey_BPI.pdf (2007)
4. Ellis, C., Nutt, G.: Workflow: The process spectrum. In Sheth, A., ed.: Proceedings of the NSF Workshop on Workflow and Process Automation in Information Systems. (May 1996) 140–145
5. Manolescu, D.A.: Micro-workflow: a workflow architecture supporting compositional object-oriented software development. PhD thesis, University of Illinois at Urbana-Champaign, Champaign, IL, USA (2001)
6. Warmer, J.B., Kleppe, A.G.: Building a flexible software factory using partial domain specific models. In: Sixth OOPSLA Workshop on Domain-Specific Modeling (DSM'06), Portland, Oregon, USA, Jyvaskyla, University of Jyvaskyla (October 2006) 15–22
7. Bennett, K.H., Rajlich, V.T.: Software maintenance and evolution: a roadmap. In: ICSE '00: Proceedings of the Conference on The Future of Software Engineering, New York, NY, USA, ACM (2000) 73–87
8. Mernik, M., Heering, J., Sloane, A.M.: When and how to develop domain-specific languages. ACM Computing Surveys **37**(4) (2005) 316–344
9. Sánchez, M., Villalobos, J.: A flexible architecture to build workflows using aspect-oriented concepts. In: AOM '08: Proceedings of the 2008 AOSD workshop on Aspect-oriented modeling, New York, NY, USA, ACM (2008) 25–30
10. Kindler, E., Axenath, B., Rubin, V.: Amfibia: A meta-model for the integration of business process modelling aspects. In Leymann, F., Reisig, W., Thatte, S.R., van der Aalst, W.M.P., eds.: The Role of Business Processes in Service Oriented Architectures. Volume 06291 of Dagstuhl Seminar Proceedings., Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), S chloss Dagstuhl, Germany (2006)
11. Sánchez, M., Villalobos, J., Romero, D.: A state machine based coordination model applied to workflow applications. Avances en Sistemas e Informática **6**(1) (June 2009)
12. Sánchez, M., Villalobos, J., Romero, D.: A state machine based coordination model applied to workflow applications. In: IV Congreso Colombiano de Computación 2009, Sociedad Colombiana de Computación (April 2009)
13. Sánchez, M., Jiménez, C., Villalobos, J., Deridder, D.: Building a multimodeling framework using executable models. In Oriol, M., Meyer, B., eds.: TOOLS EUROPE 2009. Volume 33 of LNBIP., Berlin - Heidelberg, Springer-Verlag (July 2009) 157–174
14. Sánchez, M., Villalobos, J., Deridder, D.: Co-Evolution and consistency in workflow-based applications. In: MCCM'08 Proceedings. (October 2008)
15. Sánchez, M., Barrero, I., Jorge, V., Deridder, D.: An execution platform for extensible runtime models. Technical Report COMP-005-2008, Lancaster University (2008)
16. van der Aalst, W., ter Hofstede, A.: Yawl: Yet another workflow language (revised version). Technical report, Queensland University of Technology, Brisbane (2006) QUT Technical report, FIT-TR-2003-04.

17. van der Aalst, W.M.P., Ter, Kiepuszewski, B., Barros, A.P.: Workflow patterns. Distributed and Parallel Databases **14**(1) (2003) 5–51
18. Casallas, R., López, N., Correal, D.: Eleggua: An event infrastructure for application cooperation. In Turowski, K., Zaha, J.M., eds.: COEA. Volume 70 of LNI., GI (2005) 109–123
19. Sánchez, M., Zambrano, E., González, O., López, N.: Potenciando la unión entre workflows y soluciones eai. In: II Congreso Colombiano de Computación, 2007, Sociedad Colombiana de Computacion (April 2007)
20. Agrawal, A., Amend, M., Das, M., Ford, M., Keller, C., Kloppmann, M., König, D., Leymann, F., Müller, R., Pfau, G., Plösser, K., Rangaswamy, R., Rickayzen, A., Rowley, M., Schmidt, P., Trickovic, I., Yiu, A., Zeller, M.: Ws-bpel extension for people (bpel4people), version 1.0, june 2007 (2007)
21. Wassermann, B., Emmerich, W., Butchart, B., Cameron, N., Chen, L., Patel, J.: Sedna: A bpel-based environment for visual scientific workflow modeling. In: Workflows for e-Science. Springer, London (2007) 428–449
22. Charfi, A., Mezini, M.: Aspect-oriented web service composition with ao4bpel. In: 2nd European Conference on Web Services (ECOWS). Volume 3250 of Lecture Notes in Computer Science., Berlin / Heidelberg, Springer (2004) 168–182
23. Braem, M., Verlaenen, K., Joncheere, N., Vanderperren, W., Van Der Straeten, R., Truyen, E., Joosen, W., Jonckers, V.: Isolating process-level concerns using padus. In Dustdar, S., Fiadeiro, J.L., Sheth, A.P., eds.: Business Process Management. Volume 4102 of Lecture Notes in Computer Science., Springer (2006) 113–128
24. Muth, P., Weißenfels, J., Gillmann, M., Weikum, G.: Workflow history management in virtual enterprises using a light-weight workflow management system. In: Proceedings of the Ninth International Workshop on Research Issues on Data Engineering: Information Technology for Virtual Enterprises (RIDE 99). Volume - of Lecture Notes in Computer Science., Sydney, Australia, IEEE (1999) 148–155
25. Hagen, C., Alonso, G.: Beyond the black box: event-based inter-process communication in process support systems. In: Proceedings of the 19th IEEE International Conference on Distributed Computing Systems, Washington, DC, USA, IEEE Computer Society (1999) 450–457
26. Ferreira, D.M., Ferreira, P.J.J.: Developing a reusable workflow engine. Journal of Systems Architecture: the EUROMICRO Journal **50**(6) (June 2004) 309–324
27. Fernandes, S.M., Cachopo, J., Silva, A.R.: Supporting evolution in workflow definition languages. In: SOFSEM 2004: Theory and Practice of Computer Science. Volume 2932 of Lecture Notes in Computer Science. Springer, Berlin / Heidelberg (February 2004) 49–59
28. Karsai, G., Maroti, M., Ledeczi, A., Gray, J., Sztipanovits, J.: Composition and cloning in modeling and meta-modeling. IEEE Transactions on Control System Technology (special issue on Computer Automated Multi-Paradigm Modeling **12**(2) (March 2004) 263–278
29. Brooks, C., Cheng, C., Feng, T.H., Lee, E.A., von Hanxleden, R.: Model engineering using multimodeling. In: 1st International Workshop on Model Co-Evolution and Consistency Management (MCCM '08). (October 2008)