Validación estática de compatibilidad de procesos de negocio en coreografías de servicios web

Agustín Cernuda del Río, Jose Emilio Labra Gayo
Departamento de Informática
Universidad de Oviedo
Facultad de Ciencias – C/ Calvo Sotelo, S/N - 33007 Oviedo
{guti, labra}@lsi.uniovi.es

Resumen

Muchas tecnologías asociadas a los componentes software tienen un propósito eminentemente descriptivo. Sería deseable promover estrategias de especificación y validación automática, desarrollando técnicas que a partir de las descripciones detecten defectos de manera estática (antes de la ejecución). Además, la transferencia tecnológica es un factor primordial para posibilitar la adopción de tales tecnologías. Es necesario incorporar también tales validaciones al proceso de diseño y desarrollo, a fin de avanzar en la línea del software *correcto por construcción*.

Los recientes trabajos sobre coreografías de servicios web pueden combinarse con técnicas de verificación sencillas y viables, de modo que sea posible aprovechar la información descriptiva de las coreografías para realizar el análisis descrito mediante sistemas relativamente fáciles de construir.

1. Introducción

Las iniciativas de componentes software como COM, CORBA, JavaBeans o los servicios web resolvieron problemas básicos de infraestructura para la interoperabilidad, pero no llegaron muy lejos en la verificación estática de compatibilidad entre componentes [3]. La verificación posible se basa en la comprobación de compatibilidad entre signaturas, algo que ya se recogía en las iniciativas pioneras en ese campo, como OSF-DCE [12], gracias al lenguaje IDL.

En trabajos anteriores, hemos desarrollado un modelo de verificación estática de componentes software, Itacio, que permite denotar cualquier aspecto de un componente que sea susceptible de ser expresado mediante Programación Lógica con Restricciones (en adelante CLP) y utilizar tal descripción en un sistema de verificación automática [4]. De este modo, se pueden verificar aspectos de la compatibilidad entre componentes que el sistema de tipos, por sí solo, no puede verificar. Siendo un requisito la versatilidad del modelo, se ha aplicado con éxito Itacio en campos tan diversos como la verificación de contratos de reutilización [5], la construcción de procesadores de sonido basados en componentes [6] o la verificación de requisitos de fiabilidad [7].

A un nivel mucho más alto y de grano más grueso, considerando a las organizaciones como componentes, cabe incluir en la discusión sobre componentes y compatibilidad las *Service-Oriented Architectures* (SOA). Existe una gran actividad en el desarrollo de estándares para describir estas interacciones, con una evolución parecida a la descrita para los componentes de bajo nivel: hay cierto consenso y grado de adopción en los estándares básicos de interconexión (HTTP, XML, SOAP, WSDL...) pero el panorama está menos claro en las capas superiores, donde se han sucedido diversas propuestas (XLANG, BPML, WSCL, WSCI) que parecen estar madurando en dos caminos principales: BPEL4WS ([16], [10], [8]) y WS-CDL [15].

Como es lógico, se están resolviendo en primer lugar los fundamentos de la notación y la expresividad, a partir de los casos de uso; en el desarrollo de los estándares no se está haciendo hincapié en la verificación estática. En este artículo se propone una vía de actuación para realizar verificaciones de compatibilidad de protocolos a partir de especificaciones WS-CDL y/o BPEL4WS, o bien para guiar el proceso de desarrollo con esas notaciones basándose en tales verificaciones de compatibilidad.

En primer lugar se realizará una breve descripción de las tecnologías WS-CDL y BPEL4WS. Posteriormente, se resume el marco teórico de la verificación que se pretende incorporar a esas tecnologías (el modelo de Yellin y Strom). A continuación, se discuten aspectos concretos de dicha incorporación y detalles de implementación. Se aborda entonces la posibilidad de utilizar la verificación por anticipado y de manera preventiva, como base de una metodología de diseño de procesos de negocio y no como medio para detectar defectos; y se ofrecen, finalmente, unas conclusiones.

2. Integración de servicios web

2.1. Coreografías de servicios web

WS-CDL 1.0 es, en este momento, un borrador de trabajo desarrollado por un grupo del W3C. A la espera de la versión definitiva, cabe adelantar que su propósito ([15], 1.2, 1.3) es describir de forma precisa colaboraciones entre partes independientemente de los respectivos detalles de implementación, por lo que la verificación estática no se contempla como uno de sus fines principales, sin perjuicio de que con base en la especificación se desarrollen tales mecanismos. Un ejemplo de trabajo en esta línea, con un enfoque distinto al presentado aquí, puede verse en [2], donde se formaliza WSCI mediante un álgebra de procesos (CCS) que permite razonar sobre compatibilidad de servicios web; la verificación propuesta en el presente artículo utiliza un formalismo diferente.

El modelo de WS-CDL es relativamente amplio, pero aquí cabe señalar a modo de resumen que una coreografía pretende describir la colaboración entre las partes que interactúan, en términos de las mutuas invocaciones de servicios web, intercambios de información, sincronización, etc. Dado este modelo, que todas las partes pueden acordar, cada organización puede desarrollar de forma independiente su propio papel en la colaboración, teniendo en cuenta sus particulares intereses de implementación; basta con que respete el "contrato global" que supone la coreografía descrita para que se mantenga la interoperabilidad.

La organización de las operaciones (actividades) de una coreografía se basa en las estructuras *secuencial*, *paralela* y *alternativa*, como se aprecia en la Figura 1. En esa misma ilustración se muestra también la *interacción*, una actividad elemental de especial importancia, por cuanto recoge el intercambio de información entre las partes.

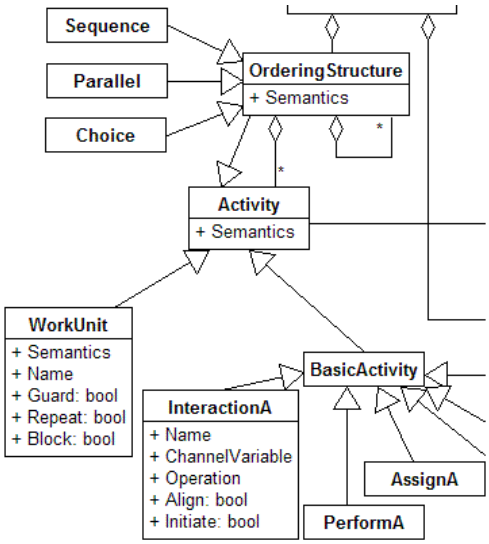

Figura 1. Entidades del metamodelo de WS-CDL implicadas en la secuenciación de actividades

2.2. Orquestación de servicios web

BPEL4WS, por su parte, permite describir un proceso de negocio de manera unilateral, en contraste con el planteamiento global y coordinado de WS-CDL. En BPEL4WS se puede abordar:

- La descripción de procesos ejecutables (es posible, con las herramientas adecuadas, traducir directamente una especificación BPEL4WS a una implementación)
- La descripción de procesos abstractos no ejecutables

Por lo que se refiere al primer punto, BPEL4WS tiene como fin definir un nuevo servicio web a partir de servicios web existentes; puede utilizarse casi como un lenguaje de programación, pero su estructura pretende plasmar la experiencia existente en integración de procesos de negocio. Esta composición de servicios web para desarrollar uno nuevo se conoce con el nombre de *orquestación* de servicios web (aunque hay opiniones diversas sobre los términos *coreografía* y *orquestación*).

En BPEL4WS las actividades primitivas son:

- Invocar una operación en un servicio web `<invoke>`
- Esperar a que una operación del servicio sea invocada desde el exterior `<receive>`
- Generar la respuesta a una operación de entrada y salida `<reply>`
- Esperar un tiempo determinado `<wait>`
- Realizar una asignación de datos `<assign>`
- Elevar excepciones `<throw>`
- Terminar la instancia del servicio `<terminate>`
- La acción vacía, no hacer nada `<empty>`

Las actividades estructurales que sirven para organizar la secuencia son:

- Secuencia ordenada `<sequence>`
- Selección múltiple `<switch>`
- Iteración `<while>`
- Ejecución de una de varias opciones, dependiendo de un evento entrante `<pick>`
- Ejecución en paralelo `<flow>`. En las actividades que se ejecutan en paralelo se pueden establecer ordenaciones parciales mediante restricciones.

El esquema general de modelado de procesos de negocio basados en servicios web (ejecutable, en mayor o menor medida, dependiendo de las herramientas utilizadas) se muestra en la Figura 2.

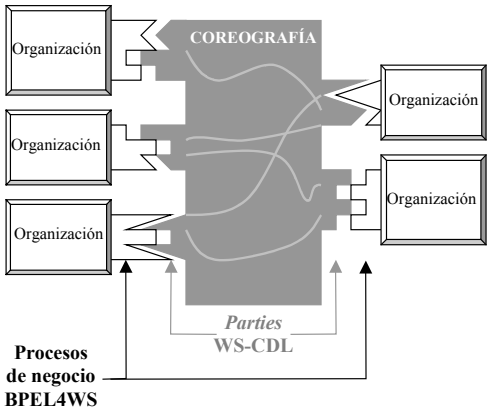

Figura 2. Esquema general de modelado de procesos de negocio basados en servicios web mediante WS-CDL y BPEL4WS

3. Compatibilidad entre protocolos

3.1. Selección del marco teórico

De cara a la verificación de compatibilidad entre protocolos, hay que señalar diversos modelos formales disponibles, como CSP o π-cálculo, que ofrecen una interesante vía de actuación [2]; de hecho, hay una clara relación entre WS-CDL y π-cálculo, aunque no se ha establecido tal relación en un plano formal [1]. A pesar de su robustez, los modelos formales en general afrontan dificultades de adopción y transferencia tecnológica. Bien porque su fuerte carga teórica así lo justifique o bien porque, en cualquier caso, tal sea la percepción de las empresas [9], muchas organizaciones son recelosas ante lo que consideran técnicas poco prácticas o que, para poder explotar adecuadamente, requieren disponer de un personal con una formación muy específica.

Nuestra propuesta de investigación, complementaria a las anteriores, parte del modelo propuesto por Yellin y Strom [17] (en adelante YS). Estos autores definen una notación para la especificación de protocolos síncronos relativamente sencilla, basada en una notación de máquina de estados. Dadas dos especificaciones, se definen algoritmos que detectan dos tipos de defectos: *recepciones no especificadas* (es decir, posibles invocaciones imprevistas por el receptor) e *interbloqueos*.

En investigaciones anteriores, hemos podido comprobar que tanto la descripción de los protocolos como el algoritmo de verificación de compatibilidad se pueden implementar sobre un motor de inferencia basado en CLP [3]. La descripción de los protocolos es declarativa y su traducción a cláusulas Horn es prácticamente directa, y algo similar ocurre con el algoritmo de YS para la detección de incompatibilidades.

3.2. El modelo de Yellin y Strom

En [17] se define un modelo (en adelante modelo YS) para complementar la especificación de las interfaces de las aplicaciones mediante restricciones de secuencia llamadas *protocolos*. Un protocolo hace explícitas las relaciones entre los mensajes (métodos) de la aplicación. Puede verse una clara similitud entre este propósito y el de las coreografías de servicios web.

Lo que se comprueba aquí es que dos componentes son *compañeros válidos*, en el sentido de que sus respectivos protocolos de llamada definen una interacción libre de cierto tipo de errores que un sistema de verificación de tipos, por sí solo, no puede detectar. Para los protocolos incompatibles se puede derivar un adaptador que permita su interacción.

El algoritmo de verificación de compatibilidad está diseñado de modo que no requiere tecnologías complejas como las de verificación de modelos, sino que puede implementarse sobre muchos lenguajes y entornos (incluyendo un motor de inferencia de lógica de primer orden, como se ha dicho).

En el modelo YS las colaboraciones tienen lugar siempre entre dos partes. Esto puede parecer una limitación, pero no hay tal, ya que es posible modelar colaboraciones complejas reduciéndolas a sus componentes bilaterales. Además, en el caso que nos ocupa se busca precisamente estudiar la compatibilidad entre la especificación de cada parte y la especificación de la coreografía, por lo que hay dos operandos en la verificación.

Un componente expone una interfaz para cada patrón de interacción, y dicha interfaz lleva asociada una *especificación de colaboración*. Esta consta de dos partes: la *signatura de la interfaz*, que describe el conjunto de mensajes que se pueden intercambiar (pudiendo ser cada mensaje *enviado* o *recibido*), y el *protocolo*, que describe un conjunto de *restricciones de secuencia* mediante una gramática de estados finitos. Basta describir el protocolo como una máquina de estados en la que los eventos que producen una transición son siempre la recepción o el envío de un mensaje.

Se supone, además, que un componente que se encuentra en un estado en el que pueda enviar un mensaje no permanecerá en él indefinidamente, sino que en algún momento lo enviará. Con esto se descarta la posibilidad de que el sistema permanezca bloqueado sin motivo.

Cabe mencionar que el modelo YS se ciñe al caso de una comunicación *síncrona*, lo cual permite razonar con mayor facilidad sobre los sistemas implicados.

Finalmente, nótese que el trabajo de Yellin y Strom se centra en su segunda parte en la construcción de adaptadores para hacer compatibles protocolos inicialmente incompatibles, aspecto que puede tener gran interés pero que queda fuera del propósito de este artículo.

4. De la especificación a la verificación

4.1. Planteamiento

El núcleo del trabajo es establecer una correspondencia entre las estructuras de control y declaraciones de interfaces de los modelos de negocio (tanto a nivel de coreografía como de orquestación), por un lado, y la notación YS, por otro (es decir, máquinas de estados). De este modo es posible aplicar la teoría de Yellin y Strom a la verificación de compatibilidad entre las diversas orquestaciones (los procesos de negocio particulares de cada parte) y la coreografía global (el acuerdo común de interacción). Siendo WS-CDL una notación declarativa, parece viable tal correspondencia; lo mismo cabe decir respecto a las orquestaciones denotadas con BPEL4WS, ya que un proceso BPEL4WS es, básicamente, la expresión de un algoritmo como diagrama de flujo [16]. El planteamiento se describe en la Figura 3.

Tras traducir una especificación WS-CDL a YS, se tomarán especificaciones del comportamiento de las partes, bien expresadas directamente en YS o bien extrayéndolas de especificaciones como BLEP4WS (que es lo que se supone en este artículo), dependiendo de lo que resulte más adecuado en el contexto empresarial. Finalmente, resultará posible verificar la conformidad entre cada proceso de negocio y la coreografía global definida, ya mediante el propio algoritmo de YS o mediante variantes *ad hoc*.

4.2. Coreografía

Como se ha dicho en 2.1, en WS-CDL al nivel más básico las actividades se organizan según estructuras *secuencial*, *paralela* y *alternativa*, además de existir otras construcciones de nivel superior (WorkUnit).

Las estructuras secuencial y alternativa no plantean mayores problemas, pero la estructura paralela es una fuente tradicional de dificultades cuando se trata con máquinas de estados [13]. La expansión de estructuras paralelas de WS-CDL en la notación YS podría acarrear una explosión del espacio de estados. En este caso, se pueden aplicar técnicas que mitigan tal problema, como factorizar el espacio de estados de modo que la compatibilidad se verifique a diversos niveles de abstracción; esto reduce drásticamente el número de estados [14], evitando examinar transiciones *internas*.

Por lo demás, es necesario definir una correspondencia detallada entre las interacciones, con sus posibles variantes recogidas en la especificación [15], y los mensajes enviados y recibidos según la terminología YS. De las demás actividades, algunas (como *NoActivity*) son internas y no tienen un papel preponderante en el protocolo.

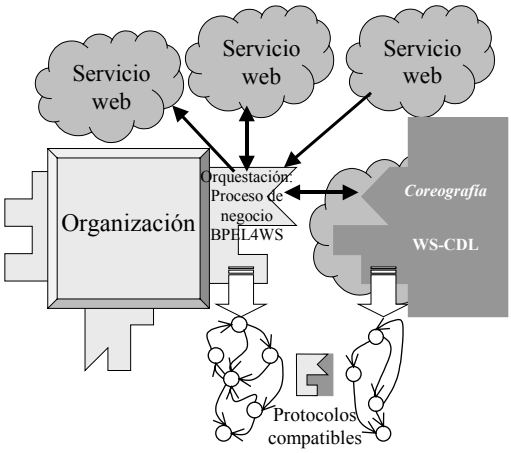

Figura 3. Esquema del proceso de verificación, obteniendo representaciones de los procesos de negocio en formato YS

4.3. Orquestación

Respecto a la orquestación, es decir, al modelado de los procesos de negocio por parte de cada entidad, cabe señalar algunos detalles de importancia.

El planteamiento general es similar al de las coreografías. Dejando aparte en una primera fase cuestiones complejas como el tratamiento de errores, básicamente se tienen:

- Actividades primitivas como `<invoke>`, `<receive>` o `<reply>` que encajan bastante bien con los *mensajes* de YS.
- Actividades irrelevantes de cara a la compatibilidad de protocolos, como `<empty>`.
- Actividades estructurales, como `<sequence>`, `<switch>` o `<while>`, que admiten una traducción sencilla al modelo YS.
- Otras actividades estructurales, como `<pick>` o `<flow>`, que pueden plantear dificultades para su encaje en el modelo de procesos YS.

La estructura `<pick>` remite, en primer lugar, a la posibilidad de transiciones no deterministas. Esto no está contemplado en el modelo YS, de modo que un camino sería estudiar una ampliación del mismo, pero parece más sencillo representar el evento entrante que determina la transición de una estructura `<pick>` como un mensaje más del modelo YS.

La estructura `<flow>` reproduce un problema ya descrito al discutir el tratamiento de las coreografías: la ejecución en paralelo, que no está contemplada en el modelo YS original. El desarrollo de las estructuras paralelas para su representación mediante máquinas de estados puede traer problemas prácticos si la explosión del espacio de estados supera las capacidades de proceso de la herramienta, pero ya se ha mencionado que existen técnicas para intentar paliar este problema (4.2).

En las orquestaciones hay que tener en cuenta una cuestión más: como se deduce de la Figura 3, un proceso de negocio BPEL4WS invoca a muy diversos servicios web, que pueden estar fuera de la definición de la coreografía. El modelo YS, sin embargo, sólo permite realizar una verificación de compatibilidad bilateral. Por tanto, se hace necesario prever de alguna forma el tratamiento de *mensajes* que no proceden de las dos partes implicadas en una verificación.

El modelo YS tiene una ventaja en relación con este problema: contempla y formaliza el concepto de *subprotocolos*, cuya existencia cabe comparar con la existencia de *subtipos* en los sistemas de tipos.

Un protocolo P es un *subprotocolo* de P' sii se cumplen todas las condiciones siguientes:

- El estado inicial de P es el mismo que el de P'
- Todos los estados finales de P' son también estados finales de P
- P puede obtenerse de P' mediante las siguientes operaciones:
  - Añadir un conjunto S de nuevos estados a P
  - Añadir un conjunto de nuevas transiciones de recepción a P
  - Añadir un conjunto de nuevas transiciones de envío a P, de manera que cada nueva transición emane de un nuevo estado de P (de uno de los pertenecientes a S)

El lema 2.4.1 de [17] establece que si P_1 es un subprotocolo de P y P es compatible con P_2,

entonces P es compatible con P_2. El proceso de negocio completo contendrá estados, envíos y recepciones adicionales, relacionados con los servicios web ajenos a la coreografía estudiada; pero si puede denotarse como subprotocolo de un protocolo compatible con la coreografía (considerando las interacciones con los servicios ajenos como operaciones internas e irrelevantes) se habrá demostrado, en virtud del lema, la compatibilidad entre el proceso de negocio y la coreografía.

4.4. Subconjunto verificable

Se han descrito diferentes vías de solución para las posibles limitaciones que el modelo YS pueda presentar al traducir los modelos WS-CDL y BPEL4WS. Pero también cabe plantearse otra vía de actuación: la definición de un *subconjunto verificable* de ambas notaciones.

Resulta relativamente fácil oponer, ante casi todas las tecnologías de análisis estático a cualquier nivel, un contraejemplo elaborado que no resulte fácil de procesar, o que haga aparecer los temidos problemas de indecidibilidad.

Estos contraejemplos, que constituyen un argumento adverso en el plano teórico, en muchos casos no resultan relevantes desde un punto de vista práctico. Si no se pierden de vista los objetivos reales de estas técnicas, resulta perfectamente aceptable plantear limitaciones en el uso de los lenguajes de programación, precisamente para evitar características negativas en el software que se construye. La programación estructurada, sin ir más lejos, no es más que eso.

Al hablar de notaciones para el desarrollo de procesos de negocio -técnicas que se encuentran aún en desarrollo y en período de adopción por la industria- resulta aceptable un planteamiento inicial en el cual se imponen limitaciones en el uso de tales notaciones (incluso limitaciones significativas) a cambio de la disponibilidad de verificaciones automáticas. Posteriormente, se puede ampliar la aplicación de dichas notaciones en paralelo con el desarrollo de técnicas de verificación que permitan confiar en la robustez de los nuevos usos.

4.5. Cuestiones de implementación

Ya se ha mencionado que en [3] se ensayó con éxito una representación de YS en declaraciones CLP. De cara a la implementación y a la fácil adopción de esta técnica, parece viable transformar directamente las especificaciones WS-CDL y BPEL4WS en representaciones YS en CLP. Siendo esta una notación textual, y siendo WS-CDL (y en su caso BPEL4WS) notaciones basadas en XML, se puede describir la transformación en XSLT, como lenguaje computacionalmente completo que es.

De ese modo, la traducción a YS se realiza mediante un simple procesador de XSLT, y el proceso de verificación mediante un motor de inferencia CLP.

5. La verificación estática como fundamento de una metodología de diseño

5.1. Construcción de procesos de negocio multiparte

Ahondando en el propósito de incorporar el análisis estático encaminado a la detección de defectos como parte rutinaria del proceso de desarrollo de software, cabe alterar el modelo de verificación descrito anteriormente (Figura 3) para convertirlo en proceso de *construcción* (Figura 4).

En la línea presentada en el apartado 4.4, se utilizará un *subconjunto verificable* de WS-CDL como base para la definición de la coreografía. Esto asegura la corrección estricta de los pasos siguientes del desarrollo.

Una vez definida y acordada dicha coreografía entre las entidades implicadas, podrán desarrollarse los servicios web asociados a los respectivos procesos de negocio mediante un proceso de *derivación*.

5.2. Derivación de procesos de negocio

El proceso de derivación para la construcción de los respectivos servicios web que se integran en la coreografía se basa, en primer lugar, en una importante propiedad del modelo YS. Dado un protocolo P, se define el *protocolo inverso* $\overline{P}$ como el derivado de P invirtiendo la dirección de todos los mensajes, de modo que los enviados pasen a ser recibidos y viceversa.

Aunque con una semántica asíncrona no podría asegurarse, con la semántica síncrona del modelo YS *el protocolo inverso es siempre compatible con el protocolo original*. Esto nos da

un punto de partida para la definición del protocolo asociado al proceso de negocio de cada parte.

Dado el protocolo inverso del que se parte, se puede construir el proceso de negocio completo garantizando que sea siempre un *subprotocolo* del inverso (se han resumido las condiciones que debe cumplir un subprotocolo en la sección 4.3). Haciendo esto, en todo momento se tiene la seguridad de que los procesos de negocio son compatibles a nivel de protocolo con la coreografía.

Disponiendo de una correspondencia clara entre el modelo YS y BPEL4WS, no será difícil desarrollar la especificación ejecutable del proceso sin romper en ningún momento la compatibilidad con la coreografía.

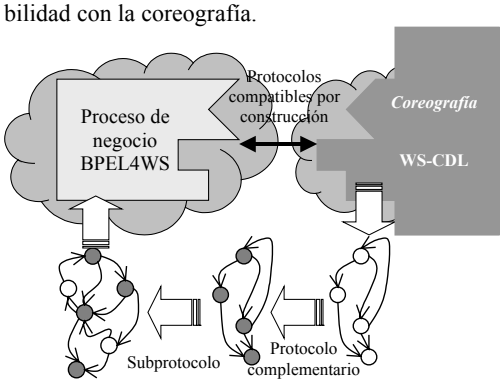

Figura 4. Metodología de desarrollo de procesos de negocio compatibles, por construcción, con la coreografía acordada

6. Conclusiones

Parece viable abordar el desarrollo de un marco de verificación que permita complementar la capacidad expresiva de WS-CDL y BPEL4WS, aportando garantías adicionales sobre la robustez de los sistemas desarrollados y mediante recursos tecnológicos ampliamente disponibles. En concreto, aquí se ofrece una discusión sobre la compatibilidad a nivel de protocolos, tal como la definen Yellin y Strom, lo que permite evitar invocaciones imprevistas e interbloqueos.

Las nociones de protocolo inverso y subprotocolo del modelo YS ofrecen, además, un punto de partida para la definición de una metodología rigurosa de desarrollo de procesos de negocio multiparte (coreografías) que aporta un alto grado de confianza en la compatibilidad de protocolos, a pesar del trabajo autónomo de las diversas organizaciones que se pretende promover gracias a la independencia que permite la especificación de un marco global mediante WS-CDL. Lo que se está proponiendo es, pues, una metodología de desarrollo de software *correcto por construcción*, concretamente a nivel de SOA.

Todo lo anterior ahonda en la idea de incorporar tecnologías de análisis estático que permitan la verificación automática del trabajo realizado y la detección temprana de defectos, ya sea sobre especificaciones existentes o mediante el enfoque de derivar las especificaciones según las pautas del modelo de verificación.

Para conseguir estos objetivos ya desde los primeros pasos en la adopción de las técnicas de descripción de procesos de negocio, parece oportuno definir un *subconjunto verificable* de las notaciones empleadas. Esto probablemente implique unas limitaciones en la expresividad, que como contrapartida permitan mayores garantías de corrección sobre el trabajo realizado; es este un intercambio muy frecuente en la aplicación de técnicas relacionadas con la calidad del software.

Agradecimientos

Este trabajo ha sido financiado por Seresco, S.A. y el Principado de Asturias en el marco de la convocatoria de Proyectos de Investigación Concertada 2004-2006 (ref. PC04-39).

Referencias

[1] Barros, A.; Dumas, M.; Oaks, P. *A Critical Overview of the Web Services Choreography Description Language (WS-CDL)*. BPTrends, 03/2005.

[2] Brogi, A.; Canal, C.; Pimentel, E.; Vallecillo, A. *Formalizing Web Service Choreographies*. Próxima aparición en Electronic Notes in Theoretical Computer Science.

[3] Cernuda del Río, A. *Sistema de verificación de componentes software* [CD]. Tesis Doctoral, 10/2002, Servicio de Publicaciones (Univ. de Oviedo). ISBN: 84-8317-316-6.

[4] Cernuda del Río, A; Labra Gayo, J. E.; Cueva Lovelle, J. M. *A Model for Integrating Knowledge into Component-Based Software Development*. 4th International ICSC Symposium - Soft Computing and Intelligent

Systems for Industry (Paisley, Escocia, 06/2001). Actas del congreso. ICSC Academic Press. ISBN: 3-906454-27-4

[5] Cernuda del Río, A; Labra Gayo, J. E.; Cueva Lovelle, J. M. *Verifying Reuse Contracts with a Component Model*. VI Jornadas de Ingeniería del Software y Bases de Datos (JISBD 2001), Almagro, Ciudad Real. Actas del congreso. ISBN: 84-699-6275-2

[6] Cernuda del Río, A; Labra Gayo, J. E.; Cueva Lovelle, J. M. *Applying the Itacio Verification Model to a Component-Based Real-Time Sound Processing System*. (Constraint) Logic Programming and Software Engineering, 17th International Conference on Logic Programming (CLPSE'01 / ICLP'01), Pafos, Chipre, 12/2001.

[7] Cernuda del Río, A; Labra Gayo, J. E.; Cueva Lovelle, J. M. *Aplicación del modelo de verificación de componentes Itacio a una teoría sobre la fiabilidad de los componentes software*. IDEAS 2002 - 5º Workshop Iberoamericano de Ingeniería de Requisitos y Ambientes Software, La Habana, 04/2002.

[8] Duftler, M.; Khalaf, R. *Business Process with BPEL4WS: Learning BPEL4WS*, Part 3. IBM developerWorks, 10/2002.

[9] Eickelmann, N. *Entrevista a David Lorge Parnas*. Software Engineering Notes, vol. 24 nº 3, ACM Press, 05/1999.

[10] Khalaf, R. *Business Process with BPEL4WS: Learning BPEL4WS*, Part 2. IBM developerWorks, 08/2002.

[11] Labra Gayo, J. E. *¿Hay Lógica en la situación actual de las titulaciones informáticas?* Actas de JENUI 2004 - X Jornadas de Enseñanza Universitaria de la Informática. 07/2004, Alicante, España. Ed. Thomson. ISBN: 84-9732-334-3.

[12] The Open Group. *DCE Today*. Julio de 1998, ISBN: 1-85912-157-8

[13] Plasil, F.; Visnovsky, S. *Behavior Protocols for Software Components*. IEEE Transactions on Software Engineering, Vol. 28, nº 9, 09/2002.

[14] Visnovsky, S. *Modeling Software Components Using Behavior Protocols*. Tesis Doctoral, Charles University (República Checa), 2002.

[15] *Web Services Choreography Description Language Version 1.0* - W3C Working Draft. W3C, 17/12/2004. http://www.w3.org/TR/2004/WD-ws-cdl-10-20041217/

[16] Weerawarana, S.; Curbera, F. *Business Process with BPEL4WS: Understanding BPEL4WS*, Part 1. IBM developerWorks, 08/2002.

[17] Yellin, D. M.; Strom, R. E. *Protocol Specifications and Component Adaptors*. ACM Transactions on Programming Languages and Systems, Vol. 19, Nº 2, Marzo de 1997.