LAS ONTOLOGÍAS EN LA INGENIERÍA DE *SOFTWARE:* UN ACERCAMIENTO DE DOS GRANDES ÁREAS DEL CONOCIMIENTO

Carlos Mario Zapata Jaramillo*
Gloria L. Giraldo**
Germán A. Urrego Giraldo***

Recibido: 14/04/2009
Aceptado: 07/05/2010

RESUMEN

Los conceptos ontológicos se suelen acercar más a la ingeniería del conocimiento, por lo que los ingenieros del *software* no los suelen aplicar para resolver problemas de su área. Es necesario que los ingenieros de *software* se apropien de las ontologías, pues éstas proporcionan un vocabulario común, que podría contribuir en la solución de problemas recurrentes en ingeniería del *software*, tales como la dificultad de la comunicación entre analista e interesado para definir los requisitos de un sistema, la baja reutilización de componentes y la escasa generación automática de código, entre otros. En este artículo se presenta un primer enlace entre las ontologías y la ingeniería de *software* mediante la recopilación y análisis de la literatura relativa a la utilización de las ontologías en las diferentes fases del ciclo de vida de un producto de *software*.

Palabras clave: ontologías, ingeniería de *software*, ciclo vida del *software*, análisis, diseño, implementación y pruebas, mantenimiento.

* Ph. D en Ingeniería, profesor asociado de la Universidad Nacional de Colombia, líder del grupo de investigación en Lenguajes Computacionales. Facultad de Minas, Escuela de Sistemas. Universidad Nacional. E-mail: cmzapata@unal.edu.co
** Doctora en Informática, Grupo de Investigación en Lenguajes Computacionales. Escuela de Sistemas. Facultad de Minas. E-mail: glgiraldog@unalmed.edu.co
*** Doctor en Informática. Departamento de Ingeniería de Sistemas. Universidad de Antioquia. E-mail: gaurrego@udea.edu.co

ONTOLOGIES IN *SOFTWARE* ENGINEERING: APPROACHING TWO GREAT KNOWLEDGE AREAS

ABSTRACT

Ontology concepts have been traditionally linked to knowledge engineering and software engineers have not applied them to solve problems of this area. It is necessary that software engineers use these ontologies, since they provide a common language, which can contribute to the solution of some common software engineering problems like difficulties in communication between the analyst and the interested person in order to define a system requirements, the low components re-use, and scarce automatic generation in code generation, among others. In this paper, a first encounter between ontologies and software engineering by means of a state-of-the-art analysis related to the use of ontologies in several phases of software development life cycle is presented.

Key words: Ontologies, software development lifecycle, software engineering, knowledge engineering.

INTRODUCCIÓN

La ingeniería de *software* (IS) es un enfoque sistemático del desarrollo, operación y mantenimiento del *software* cuyos objetivos, entre otros, son mejorar la calidad de los productos de *software* y suministrar a los desarrolladores las bases para construir *software* de alta calidad en una forma eficiente La ingeniería del conocimiento (IC), por su parte, es una disciplina moderna que forma parte de la inteligencia artificial (IA) y cuyo objetivo es extraer, articular y computarizar el conocimiento de un experto [1]. La IS y la IC tienen muchos tópicos en común [2], pues ambas tratan con el modelado de objetos del mundo real. El término "ontología" proviene de la filosofía y es una especificación explícita y formal de una conceptualización compartida [3].

Actualmente, se empieza a reconocer que las ontologías pueden ayudar en la solución de problemas de la IS. En cada una de las fases del ciclo de vida de los productos de *software*, algunos trabajos de aplicación de ontologías se están llevando a cabo, por lo cual en este artículo se realiza una recopilación de aplicaciones de las ontologías en la IS, separándolas en las diferentes fases del ciclo de vida del *software*.

La estructura de este artículo es la siguiente: en la sección dos se muestran algunos antecedentes de las ontologías en diferentes áreas del conocimiento, en la sección tres se explora el uso de las ontologías en las diferentes fases del ciclo de vida del *software* y en la sección cuatro se presentan las conclusiones y el trabajo futuro.

1 ANTECEDENTES

Una de las dimensiones del marco para la clasificación de las ontologías se agrupa en tres áreas [4]: asistencia en la comunicación entre agentes humanos, logro de la interoperabilidad entre sistemas de información y mejoramiento de la calidad de los sistemas de *software*. Relativo a los beneficios de las ontologías en los sistemas de *software*, se destaca que posibilitan la capacidad de reutilización y generan confiabilidad en los sistemas, pues permiten automatizar el chequeo de la consistencia [4]. Los sistemas que usan ontologías en su construcción sirven para mejorar la documentación del *software* y así reducir costos de mantenimiento.

2 USO DE ONTOLOGÍAS EN LAS FASES DEL CICLO DE VIDA DEL *SOFTWARE*

Algunos trabajos proponen ontologías para ayudar, de manera transversal, en todo el proceso de desarrollo de *software*. En uno de ellos [5] se propone una ontología de IS, que contiene los conceptos para representar y comunicar el conocimiento en IS y la información de los proyectos de *software*. La función de esta ontología es facilitar el entendimiento común del conocimiento a los miembros de un equipo de desarrollo de *software*. En otro trabajo, Mendes y Abran [6] exploran SWEBOK (el cuerpo de conocimiento de la IS, que la delimita la IS y la organiza en una taxonomía a muy alto nivel) y proponen una ontología que aprovecha todo el conocimiento ya validado por los expertos de SWEBOK, para enriquecerlo y mejorar su estructura. Así, esta ontología integra un conocimiento detallado para apoyar cada fase del ciclo de vida del *software*.

2.1 Definición y análisis

Son las fases iniciales del ciclo de vida del *software*, en las que se realiza un conjunto de procesos que parten de la captura que hacen los analistas de los requisitos de los interesados, hasta su especificación en lenguajes formales y semiformales para el desarrollo de *software*. En estas fases, los problemas se suelen asociar con la escasa comprensión que

tienen interesados y analistas de los requisitos, especialmente por problemas de comunicación y falta de claridad en los requisitos.

2.1.1 *Uso de ontologías existentes*

Dos de estos trabajos emplean ontologías generales. El primero [7] establece que los modelos de características, que son jerarquías de características que incluyen una variabilidad, son vistas de una ontología general y utilizan estos modelos para la generación e integración de vistas. El segundo trabajo, denominado CM-Builder [8], emplea un modelo del mundo, expresado en forma de una ontología general, para clasificar los conceptos y relaciones que se incluyen en la descripción textual de los requisitos de una aplicación y luego obtener el diagrama de clases de UML.

Otros trabajos emplean ontologías del dominio. Se destacan Kaiya y Saeki [9]USA</pub-location><urls></urls></record></Cite></EndNote>, que proponen una estructura para las ontologías del dominio, correspondiente al proceso de captura de requisitos, e incorporan las ontologías así definidas en un método para verificar la completitud y consistencia de los requisitos, medir la calidad de la especificación en relación con su significado y predecir cambios en los requisitos. Soares [10] propone un conjunto de ontologías del dominio de asuntos sociales y organizacionales, que se pueden usar en las fases de análisis y diseño del desarrollo de *software*. Jin *et al.* [11] proponen el uso de una ontología de negocios para construir, en un lenguaje controlado, exento de términos de *software*, los requisitos del interesado. Geerts y McCarthy [12] emplean una ontología del dominio empresarial, basada en el modelo REA (*Resource-Event-Agent*), para apoyar el análisis conceptual, entre otras aplicaciones. Dobson *et al.* [13] utilizan una ontología perteneciente al dominio de calidad, en sistemas orientados a servicios, para atender la especificación de requisitos, el descubrimiento de servicios y la selección, diferenciación y búsqueda de servicios. Pisanelli *et al.* [14] emplean una librería de ontologías, previamente elaboradas, pertenecientes al dominio de guías médicas, con el fin de integrar modelos conceptuales y definir estándares de representación. Finalmente, Linhalis y Moreira [15] usan un entorno basado en una ontología de componentes, con el fin de identificar componentes, parámetros, métodos y acciones en un dominio particular.

2.1.2 *Construcción de ontologías como productos intermedios*

En este campo se encuentran trabajos para extraer, modelar y analizar requisitos de seguridad en la construcción de un sistema informático [16], generar planes de procesos de *software* [17], formalizar los requisitos (mediante una ontología paralela a la especificación) [18], para colaborar en la identificación de elementos de un reporte técnico de accidente de tránsito [19], generar modelos ejecutables de componentes [20], participar en el proceso de desarrollo de aplicaciones Web [21]%B, determinar el significado de un problema en el nivel de negocio [22] y participar en el proceso de traducción de descripciones textuales a diagramas de casos de uso [23].

2.1.3 *Identificación de términos relevantes de un dominio*

En este grupo de trabajos, se destaca el uso de la minería de textos como técnica para la extracción de términos. Dittenbach *et al.*[24] proponen la identificación de términos importantes de un dominio, con el fin de emplearlos luego en una ontología que represente ese dominio. Benaroch [25] presenta un método para capturar los requisitos y especificarlos de manera declarativa, haciendo explícita una ontología local que se puede traducir luego a una base de datos o a un esquema relacional. Gangemi

et al. [26] proponen un método para la integración y recopilación de términos médicos en una ontología del dominio, para apoyar procesos de ingeniería de requisitos en ese dominio.

2.2 Diseño

En esta fase, mediante un proceso iterativo, se traducen los requisitos y especificaciones de las fases previas en una representación del *software* por construir, que incluye los datos, la arquitectura, las interfaces y los procedimientos. Las ontologías contribuyen en las diversas técnicas asociadas con el diseño de *software*.

2.2.1 *Uso de ontologías existentes*

Parrend y David [27] presentan un proceso basado en ontologías del dominio para apoyar los procesos de ingeniería basada en modelos. Algo similar propone Pahl [28] para el diseño de servicios web. Otros usos de las ontologías existentes se enfocan en la recopilación de requisitos de diferentes dominios como los modelos de procesos [29] y los componentes de *software* [30]. Además, Chitchyan *et al.* [31] las emplean en el diseño de aplicaciones orientadas a aspectos y Ferreiro *et al.* [32] en la construcción de bases de datos desde documentos de la Web.

2.2.2 *Construcción de ontologías de diseño*

Para este uso en particular, Devedzic [33] propone la construcción de ontologías a partir de patrones de diseño y Romay y Cuesta [34] proponen un enfoque basado en aspectos para la construcción de ontologías del dominio durante el desarrollo de sistemas de información.

2.3 Implementación

La generación automática de código, como una solución para cerrar la brecha existente entre las etapas de análisis y diseño a la implementación, suscita mucho interés en los investigadores de IS. Las ontologías se usan, en este contexto, de diversas formas.

2.3.1 *Generación automática de código*

Bures *et al.* [35] definen la síntesis de programa (*Program synthesis*) como el proceso para derivar, automáticamente, código ejecutable desde especificaciones de alto nivel no ejecutables. Este proceso se basa en esquemas que representan el conocimiento computacional reutilizable y usa técnicas de IA. Los sistemas AutoBayes y AutoFilter, desarrollados en el centro de investigación AMES de la NASA, generan código a partir de modelos estadísticos y de estimación de estados, respectivamente, aplicando síntesis de *software* basado en esquemas. Dado que estos sistemas se volvieron inmanejables, por su alto grado de complejidad, los autores estudian las ventajas de las ontologías en este tipo de sistemas. Ellos afirman que las ontologías actúan como documentación para los programadores, facilitan la escritura de los esquemas, controlan la interacción de esquemas, facilitan la extensión a nuevos dominios, permiten validar la salida de los esquemas, aseguran la consistencia a través del proceso y posibilitan la generación de artefactos adicionales basados en conocimiento.

2.3.2 *Aprendizaje y comprensión de los lenguajes de programación*

Sosnovsky y Gavrilova [36] proponen una ontología educacional para la enseñanza y el aprendizaje del lenguaje C. Lee *et al.* [37] desarrollaron la ontología JLOO (Java® *Learning Object Ontology*), útil en el aprendizaje del lenguaje Java®. Turner y Eden [38] abordan el problema de los lenguajes de programación desde la perspectiva de la filosofía de las ciencias de la

computación y proponen una taxonomía de abstracciones de programas, que busca distinguir los programas de otras entidades, como *hardware* y especificaciones de programas o meta programas. DOLCE (*Descriptive Ontology for Linguistic and Cognitive Engineering*) [39] es una ontología general para ayudar a estructurar el dominio de la programación y que se aplica en el campo de la neurología, para compartir y reutilizar programas de procesamiento de imágenes.

2.4 Pruebas

Bench-Capon [40] utiliza ontologías para verificar la coherencia de una base de conocimientos, proveer un medio para estructurar las pruebas y sugerir respuestas apropiadas cuando las pruebas indican que existen fallas. Looker *et al.*[41] proponen un método para medir la capacidad de un sistema para ejecutar su función de una manera confiable, en el contexto de los servicios *Web* y las arquitecturas orientadas a servicios. Se construyeron varias ontologías: descripción de los servicios Web, una extensión de un modelo general de fallas y modos de errores. Yu *et al.* [42] proponen un método para probar la capacidad que poseen los servicios *Web* para operar entre ellos. El uso de la ontología permite detectar nuevos tipos de errores, permitiendo el ingreso de las reglas correspondientes.

2.5 Mantenimiento

La comunidad de la IS comparte la idea de que lo esencial para realizar un buen mantenimiento de un sistema es conocerlo a fondo. Este conocimiento abarca la definición del sistema y los supuestos para su realización, sus componentes y sus interrelaciones, las funciones asociadas a cada componente, los requisitos funcionales y no funcionales, los detalles de la implementación del sistema, las metas y procesos organizacionales que soporta.

2.5.1 *Enfoques centrados en el conocimiento del sistema y del dominio*

La ontología de Kitchenham *et al.* [43] contiene los conceptos relevantes para la clasificación de estudios empíricos en el mantenimiento de *software*. Esta ontología comprende cuatro subontologías: de actividades de mantenimiento, de procesos de la organización, de agentes involucrados en las actividades y de productos de *software*. La subontología de procesos comprende dos divisiones: de procedimientos y de organización del proceso. Oliveira *et al.* [44] agregan una quinta subontología, correspondiente al conocimiento relacionado con el dominio de aplicación. Algunos tipos de conocimiento, implícitos en los artefactos construidos en el ciclo de vida de desarrollo de un *software*, se representan como conceptos en la ontología presentada por Deridder [45]. Entre estos conocimientos, se encuentran las conexiones entre los diferentes artefactos, el conocimiento que se pierde en los refinamientos interactivos y el conocimiento considerado como de sentido común por las partes participantes en el desarrollo. Hyland-Wood *et al.* [46] construyen una ontología de conceptos de la IS. Dichos conceptos corresponden a las componentes y metadatos del sistema y permiten la navegación sobre éste facilitando el entendimiento del *software* y su mantenimiento. April [47] formaliza una ontología de mantenimiento con base en la ontología de Ruiz *et al.* [48] y establece una relación con un conjunto de mejores prácticas contenidas en la integración de modelos de capacidad y madurez (CMMI).

2.5.2 *Enfoques orientados por la transformación de modelos*

La multiplicidad de esquemas generada en el proceso de síntesis de programas, que se presenta en el numeral 3.3.1, se mejora con la introducción de una ontología de conceptos de

dominios específicos, que guía la actualización y el cambio de los sistemas, actividades representativas de la fase de mantenimiento. Nistor [49] se ocupa de hacer extensible la síntesis de sistemas, por medio de la ontología, a nuevos dominios de aplicación.

3 DISCUSIÓN, CONCLUSIONES Y TRABAJO FUTURO

Las ontologías, como una forma de integrar y hacer explícitos los conocimientos correspondientes a cada una de las fases del ciclo de vida de desarrollo del *software*, constituyen una contribución muy importante en la profundización del conocimiento requerido en IS, para mantenerlo, adaptarlo, reutilizarlo y producir mejores aplicaciones.

En general, las fases de definición, análisis y diseño emplean ontologías existentes del dominio y construyen ontologías como productos intermedios para la transformación de modelos. También, se identifican términos del dominio y se realizan comparaciones con las bases de datos, especialmente en el tema de versionado.

En implementación, se presentaron algunos trabajos que aplican ontologías en la generación automática de código, particularmente en la técnica de síntesis de programas, en el entendimiento de la naturaleza de los lenguajes de programación y en la enseñanza/aprendizaje de estos últimos.

En la fase de pruebas de sistemas convencionales, la literatura reporta pocos casos de aplicaciones de ontologías. Bertolino [50] evidencia esta carencia, cuando califica como un desafío actual la aplicación de técnicas inteligentes en las pruebas de *software*.

La fase de mantenimiento proclama la necesidad de conocer muy a fondo el sistema que se desea mantener. Así, ella tiende a convertirse en una extensión de la fase de análisis, donde tradicionalmente se concentran los conocimientos para el desarrollo de los sistemas. Las ontologías que representan ese conocimiento apoyan de manera importante esta fase del desarrollo de *software*.

Cada una de las fases del desarrollo de *software* requiere una participación mayor de las ontologías, especialmente las fases finales. Algunos trabajos seminales a partir de esta revisión de la literatura son: la elaboración de metaontologías para la educción de requisitos, la articulación de ontologías de MDA para apoyar la transformación entre modelos, la generalización del conocimiento de los diferentes lenguajes de programación en ontologías genéricas para ese fin, la recopilación del conocimiento relativo a pruebas y mantenimiento de *software* y la continuación más detallada de la taxonomía SWEBOK para darle más características ontológicas.

REFERENCIAS

[1] R. Dieng *et al.*, *Knowledge management : méthodes et outils pour la gestion des connaissances*, 3 ed., Paris: Dunod, 2005.

[2] J. Rech, y K.-D. Althoff, "Artificial Intelligence and Software Engineering: Status and Future Trends," *Journal KI*, vol. 18, no. 3, pp. 5-11, 2004.

[3] T. R. Gruber, "A translation approach to portable ontology specifications," *Knowledge Acquisition*, vol. 5, no. 2, pp. 199-220, 1993.

[4] M. Uschold, y R. Jasper, "A Framework for Understanding and Classifying Ontology Applications," en Proceedings of the IJCAI Workshop on Ontologies and Problem-Solving Methods (KRR5), Stockholm, Sweden, 1999.

[5] P. Wongthongtham *et al.*, "Ontology-based multi-site software development methodology and tools," *Journal of Systems Architecture*, vol. 52, no. 11, pp. 640-653, 2006.

[6] O. Mendes, y A. Abran, "Issues in the development of an ontology for a emerging engineering discipline," en Proceedings of 17th International Conference on Software Engineering and Knowledge Engineering, China, 2005, pp. 139-144.

[7] K. Czarnecki *et al.*, "Feature Models are Views on Ontologies," en Proceedings of the 10th International Software Product Line Conference (SPLC'06), Baltimore, Maryland, USA, 2006, pp. 41-51.

[8] H. Harmain, y R. Gaizauskas, "CM-Builder: An Automated NL-based CASE Tool," en Proceedings of the 15th IEEE International Conference on Automated Software Engineering (ASE'00), Grenoble, Francia, 2000.

[9] H. Kaiya, y M. Saeki, "Using Domain Ontology as Domain Knowledge for Requirements Elicitation," en Proceedings of the 14th IEEE International Requirements Engineering Conference, Minnesota, USA, 2006, pp. 189-198.

[10] A. Soares, "A social and organisational ontological foundation for Enterprise Modelling," en Proceedings of the IFAC Symposium on Manufacturing Modelling, Management and Control (MIM), Patras, Grecia, 2000.

[11] Z. Jin *et al.*, "Automatically Acquiring the Requirements of Business Information Systems by using Business Ontology," en Proceedings of European Conference Artificial Intelligence (ECAI), Workshop on Applications of Ontologies and Problem-Solving Methods, Brighton, 1998.

[12] G. Geerts, y W. McCarthy, "The Ontological Foundation of REA Enterprise Information Systems," en Proceedings of the Annual Meeting of the American Accounting Association, San Francisco, CA, 2005.

[13] G. Dobson *et al.*, "Quality of Service Requirements Specification Using an Ontology," en Proceedings of the SOCCER Workshop, at Requirements Engineering Conference, Paris, 2005.

[14] D. Pisanelli *et al.*, "The Role of Ontologies for an Effective and Unambiguous Dissemination of Clinical Guidelines," *Lecture Notes in Computer Science*, vol. 1937, pp. 129-139, 2000.

[15] F. Linhalis, y D. Moreira, "Semantic Mapping between UNL Relations and Software Components to the Execution of Natural Language Requisitions," en Proceedings of the 3rd International Information and Telecommunication Technologies Symposium, São Carlos, Brasil, 2004, pp. 109-116.

[16] S. Lee *et al.*, "Building Problem Domain Ontology from Security Requirements in Regulatory Documents," en Proc. of the 2006 workshop on software engineering for secure systems, Shanghai, China, 2006, pp. 43-50.

[17] T. Yamaguchi, "Modeling Software Processes by using Process and Object Ontologies," en Proceedings of the 12th IEEE International Conference on Automated Software Engineering, Incline Village, 1997, pp. 319-320.

[18] D. Damian *et al.*, "Integration of Behavioural Requirements Specification within Compositional Knowledge Engineering," *Knowledge-Based Systems* vol. 18, no. 7, pp. 353-365, 2005.

[19] R. Johansson *et al.*, "Carsim: A System to Visualize Written Road Accident Reports as Animated 3D Scenes," en Proceedings of the ACL second Workshop on Text Meaning and Interpretation, Barcelona, España, 2004, pp. 57-64.

[20] B. Bryant *et al.*, "From Natural Language Requirements to Executable Models of Software Components," en Proc. of the Monterrey Workshop on Soft. Eng. for Embedded Systems: From Requirements to Implementation, Chicago, 2003, pp. 51-58.

[21] K. Breitman, y J. Leite, "Ontology as a Requirements Engineering Product," en Proceedings of the 11th IEEE International Conference on Requirements Engineering, Monterrey Bay, 2003, pp. 309-319.

[22] J. Barjis *et al.*, "Language Based Requirements Engineering Combined with Petri Nets," en Proceedings of Evaluation of Modeling Methods in Systems Analysis and Design (EMMSAD), Toronto, 2002.

[23] B. Shishkov *et al.*, "Using norm analysis to derive use case from business processes," en Proceedings of 5th Workshop on Organizations semiotics OS, Delft, Netherlands, 2002, pp. 187-195.

[24] M. Dittenbach *et al.*, "Improving Domain Ontologies by Mining Semantics from Text," en Proceedings of the first Asian-Pacific Conference on Conceptual Modelling, Dunedin, 2004, pp. 91-100.

[25] M. Benaroch, "Specifying Local Ontologies in Support of Semantic Interoperability of Distributed Inter-organizational Applications," en Proc. of the 5th Intl. Workshop on Next Generation Inf. Techn. and Systems, Caesarea, Israel, 2002, pp. 90-106.

[26] A. Gangemi *et al.*, "An Overview of the ONIONS Project: Applying Ontologies to the Integration of Medical Terminologies," *Data and Knowledge Engineering*, vol. 31, no. 2, pp. 183-220, 1999.

[27] P. Parrend, y B. David, "Use of Ontologies as a Way to Automate MDE Processes," en Proc. of the IEEE EuroCon Conference, Belgrade, Serbia, 2005, pp. 567-570.

[28] C. Pahl, "Semantic Model-Driven Architecting of Service-Based Software Systems," *Information and Software Technology* vol. 19, no. 8, pp. 838-850, 2007.

[29] M. Gnatz *et al.*, "Towards a Living Software Development Process based on Process Patterns," *Lecture Notes in Computer Science,* vol. 2077, pp. 182-202, 2001.

[30] H. Happel *et al.*, "KOntoR: An Ontology-enabled Approach to Software Reuse," en Proceedings of the 18th Intl. Conf. on Software Engineering & Knowledge Engineering, San Francisco, 2006, pp. 349-354.

[31] R. Chitchyan *et al.*, *Initial Version of Aspect-Oriented Requirements Engineering Model,* Deliverable D36, Document No. AOSD-Europe-ULANC-17, AOSD-Europe, University of Lancaster, 2006.

[32] J. Ferreiro *et al.*, "Generación Automática de una Base de Datos desde Documentos de la Web," en Memorias del Congreso Argentino de Ciencias de la Computación, Ushuaia, 2000.

[33] V. Devedzic, "Ontologies: Borrowing from Software Patterns," *Intelligence,* vol. 10, no. 3, pp. 14-24, 1999.

[34] M. Romay, y C. Cuesta, "Hacia la definición de Ontologías Orientadas a Aspectos," en Memorias del Taller de Desarrollo de Software Orientado a Aspectos (DSOA), Granada, España, 2005.

[35] T. Bures *et al.*, "The role of ontologies in schema-based program synthesis," en Proceedings 19th annual ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA). Workshop on Ontologies as Software Engineering Artifacts, 2004.

[36] S. Sosnovsky, y T. Gavrilova, "Development of educational ontology for c-Programming," *Intl. Journal Information Theories & Applications,* vol. 13, no. 4, pp. 303-308, 2006.

[37] M.-C. Lee *et al.*, "Java learning object ontology," en Proceedings of Fifth IEEE International Conference on Advanced Learning Technologies, ICALT, Kaohsiung, Taiwan, 2005, pp. 538-542.

[38] R. Turner, y A. H. Eden, "Towards a Programming Language Ontology," en *Computing, Information, Cognition The nexus and the liminal*, G. Dodig-Crnkovic and S. Stuart, eds., Cambridge: Cambridge Scholars Press, 2007.

[39] P. Lando *et al.*, "Towards a general ontology of computer programs," en Proceedings of 2nd International Conference on Software and Data Technologies, ICSOFT, Barcelona, España, 2007.

[40] T. J. M. Bench-Capon, "The Role of Ontologies in the Verification and Validation of Knowledge Based Systems," en 9th International Workshop on Database and Expert Systems Applications (DEXA), 1998.

[41] N. Looker *et al.*, "An Ontology-Based Approach for Determining the Dependability of Service-Oriented Architectures," en Proceedings of the 10th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems (WORDS), 2005, pp. 171-178.

[42] Y. Yu *et al.*, "Web services interoperability testing based on ontology," en Proceedings of the 5th International Conference on Computer and Information Technology (CIT), Shanghai, China, 2005, pp. 1075-1079.

[43] B. A. Kitchenham *et al.*, "Towards an ontology of software maintenance," *Journal of Software Maintenance: Research and Practice* vol. 11, no. 6, pp. 365-389, 1999.

[44] K. M. Oliveira *et al.*, "Knowledge for Software Maintenance," en Proceedings Fifteenth International Conference on Software Engineering and Knowledge Engineering (SEKE), 2003, pp. 61-68.

[45] D. Deridder, *Facilitating software maintenance and reuse activities with a concept-oriented approach*, Brussels: Programming Technology Lab, Vrije Universiteit Brussel, 2002.

[46] D. Hyland-Wood *et al.*, "Enhancing Software Maintenance by using Semantic Web Techniques," en International Semantic Web Conference (ISWC), Athens, GA, USA, 2006.

[47] A. April *et al.*, "A Formalism of ontology to support a software maintenance knowledge-based system," en Proceedings of the Eighteenth International Conference on Software Engineering & Knowledge Engineering Conference (SEKE06), San Francisco, CA, USA, 2006, pp. 331-336.

[48] F. Ruiz *et al.*, "An Ontology For The Management Of Software Maintenance Projects," *International Journal of Software Engineering and Knowledge Engineering,* vol. 14, no. 3, pp. 323-349, 2004.

[49] E. Nistor, *Using Domain Models in Extensible Schema-based Software Synthesis*; http://www.ics.uci.edu/enistor/research/nistor-report.pdf, California, 2004.

[50] A. Bertolino, "Software Testing Research: Achievements, Challenges, Dreams," en Proceedings of 29th International Conference on Software Engineering. ICSE, Minneapolis, USA, 2007, pp. 85-103.