ISBN: 978-950-42-0200-4
DOI: https://doi.org/10.33414/ajea.5.747.2020

Uso de ontologías y tecnologías de la web semántica para la integración, recuperación e intercambio de conocimiento de arquitecturas de software

Use of ontologies and semantic web technologies for integration, retrieval and sharing of software architectures knowledge

Presentación: 06/10/2020

Doctorando:

Francisco Javier Peña Veitía

Instituto de Desarrollo y Diseño, Consejo Nacional de Investigaciones Científicas y Técnicas / Universidad Tecnológica Nacional, Argentina
fpveitia@santafe-conicet.gov.ar

Director/a:

María Marcela Vegetti

Co-director/a:

María Luciana Roldán

Resumen

A medida que los sistemas de software evolucionan, sus arquitecturas de software se vuelven más complejas, y crece la cantidad de datos generados durante los procesos de desarrollo, los cuales contienen alguna forma de conocimiento arquitectónico. Si bien muchas organizaciones han adoptado algunas estrategias y herramientas para brindar soporte a los productores de estos datos, existen aún dificultades para la extracción y/o formalización del conocimiento intrínsicamente embebido en esta información, así como su uso y distribución en proyectos de software posteriores. La importancia del estudio, captura y representación del conocimiento sobre arquitecturas de software (CAS) y su gestión ha dado lugar a numerosos trabajos de investigación, algunos de ellos haciendo uso de ontologías aplicadas como herramienta de soporte al conocimiento. Con el surgimiento del concepto de decisión arquitectónica, aparecieron las primeras ideas en el empleo de ontologías aplicadas al diseño de arquitecturas de software, por ejemplo, para describir decisiones arquitectónicas y relaciones entre ellas, para registrar decisiones arquitectónicas, artefactos de software, hojas de ruta e intereses arquitectónicos. El objetivo general de la investigación abordada es la definición de metodologías, herramientas y mecanismos para explotar el

conocimiento de arquitecturas de software que se encuentra codificado en fuentes diversas y heterogéneas, haciendo uso de ontologías, tecnologías de la web semántica y algoritmos de aprendizaje automático; para facilitar el acceso, integración, recuperación, y aplicación en nuevos proyectos de diseño, así como también para inferir nuevo conocimiento a partir del existente.

Palabras clave: arquitectura de software, conocimiento arquitectónico, ontología, aprendizaje automático

Abstract

As software systems evolve and grow, their software architectures become more complex and therefore the amount of data containing some form of architectural knowledge also continues to grow. While many organizations have adopted some strategies and tools to support the producers of this data, there are still difficulties in extracting and/or formalizing the knowledge intrinsically embedded in this information, as well as its use and distribution in subsequent software projects. The importance of the study, capture and representation of knowledge about software architectures (CAS) and its management has given rise to numerous research works, some of them making use of applied ontologies as a tool to support knowledge. With the emergence of the concept of architectural decision, the first ideas appeared in the use of ontologies applied to the design of software architectures, for example, to describe architectural decisions and relationships between them, to record architectural decisions, software artifacts, roadmaps and architectural interests. The general objective of this work is the definition of methodologies, tools and mechanisms to exploit the knowledge of software architectures that is codified in diverse and heterogeneous sources, making use of ontologies, semantic web technologies and automatic learning algorithms; to facilitate access, integration, retrieval, and application in new design projects, as well as to infer new knowledge from existing knowledge.

Keywords: software architecture, architectural knowledge, ontology, machine learning

Introducción

La importancia del conocimiento sobre arquitecturas de software (CAS) y su gestión ha dado lugar a numerosos trabajos de investigación en relación a su captura y representación (Tang et al., 2010)(Capilla et al., 2016), así como a su uso para intercambiar/compartir dicho conocimiento, para validación/verificación de conformidad, descubrimiento de nuevo conocimiento y trazabilidad (sharing, compliance, discovering, traceability)(Capilla et al., 2016). Los primeros trabajos en representación de conocimiento sobre arquitecturas de software se enfocaron fundamentalmente en cómo describir las arquitecturas en términos de su estructura y comportamiento. Posteriormente, el foco de los trabajos de investigación estuvo puesto en la representación de conocimiento arquitectónico relativo a las decisiones arquitectónicas tomadas durante un proceso de diseño de arquitecturas de software y su razonamiento. Mientras que los primeros se centraron en la representación del artefacto final, los últimos lo hicieron en la representación de cómo éste había sido obtenido, mediante qué decisiones. Actualmente, es posible aseverar que el conocimiento arquitectónico abarca diferentes categorías de conocimiento: conocimiento general (por ejemplo, tipos de artefactos manejados, patrones de diseño y tácticas), conocimiento de contexto (requerimientos, restricciones), conocimiento de diseño (las diferentes descripciones arquitectónicas, vistas, modelos de una arquitectura de software obtenidas en diferentes proyectos), y conocimiento de razonamiento (las decisiones tomadas, los argumentos que las respaldaron, las alternativas que se evaluaron en los proyectos de diseño arquitectónico)

(Tang et al., 2010). Se han propuesto además diversas formas para representar tal conocimiento (Heesch et al., 2012) empleándose enfoques basados en archivos o documentos textuales (como el uso de plantillas de decisiones arquitectónicas y wikis), anotaciones (que se vinculan a descripciones arquitectónicas), y estructuración del mismo en bases de datos vinculadas a las herramientas de captura.

Una de las modalidades de representación de conocimiento arquitectónico más utilizada es la documentación basada en archivos (file-based documentation). Generalmente se la organiza en plantillas para documentar decisiones arquitectónicas, y se la complementa con otras descripciones arquitectónicas basadas en diversas vistas (Clements et al., 2010). La naturaleza "lineal" que tiene la documentación de CAS basada en archivos dificulta estructurar el conocimiento de manera que sea útil para los diferentes usuarios de la documentación, ya que organiza el conocimiento de una manera estática y lineal, siendo difícil recuperar cierto tipo de conocimiento que esa estructura no soporta (de Graaf et al., 2014).

Con el surgimiento del concepto de decisión arquitectónica, aparecieron las primeras ideas en el empleo de ontologías aplicadas al diseño de arquitecturas de software. (Kruchten et al., 2005) propusieron una ontología para describir decisiones arquitectónicas y relaciones entre ellas, incluyendo aspectos de razonamiento. Para explotar el conocimiento de la ontología propusieron además una herramienta para preservar los grafos de decisiones de diseño y todas sus interdependencias con el fin de brindar soporte a la evolución y mantenimiento de los sistemas. En la misma dirección, la propuesta de (Akerman & Tyree, 2006) apuntaba a registrar decisiones arquitectónicas, artefactos de software (software assets), hojas de ruta (roadmaps) e intereses arquitectónicos (architectural concerns).

Este trabajo se enfoca en aquellos actores que son consumidores o usuarios del conocimiento arquitectónico, buscando facilitar la recuperación del conocimiento que ha sido documentado o representado previamente. La motivación para este plan de investigación está relacionada con el hecho de que a medida que los sistemas de software evolucionan y crecen, sus arquitecturas de software se vuelven más complejas. Las organizaciones llevan a cabo numerosos proyectos y la cantidad de conocimiento arquitectónico que éstas generan también continúa creciendo. Por otro lado, el conocimiento de arquitecturas de software generado en las organizaciones constituye un capital valioso que puede ser compartido en una comunidad. Si bien muchas organizaciones han adoptado algunas estrategias y herramientas para brindar soporte a los productores de dicho conocimiento (arquitectos, diseñadores, personal a cargo de documentación y mantenimiento, etc.), existen aún dificultades para que este conocimiento pueda compartirse, ya sea dentro de una misma organización o hacia afuera, y lograr que quede disponible a los consumidores (personal de mantenimiento, desarrolladores, revisores, otras organizaciones, otras herramientas de gestión de conocimiento, etc.).

El objetivo general de la investigación abordada es la definición de metodologías, herramientas y mecanismos para explotar el conocimiento de arquitecturas de software que se encuentra codificado en fuentes diversas y heterogéneas, haciendo uso de ontologías, tecnologías de la web semántica y algoritmos de aprendizaje automático; para facilitar el acceso, integración, recuperación, y aplicación en nuevos proyectos de diseño, así como también para inferir nuevo conocimiento a partir del existente.

Desarrollo

Para alcanzar el objetivo del trabajo de investigación se propondrá una red de ontologías que posibilite la integración de conocimiento arquitectónico proveniente de diversas fuentes heterogéneas y su recuperación

de una manera eficiente. Para ello se aprovecharán las ventajas que ofrecen las ontologías y las tecnologías de web semántica. Una ontología se refiere a un modelo de dominio formal que describe los conceptos y relaciones de ese dominio (Uschold & Gruninger, 1996). Las ontologías posibilitan la clasificación jerárquica de conceptos de dominio interrelacionados y pueden ser representadas empleando un esquema RDF[1] o el lenguaje OWL[2]. El uso de estos lenguajes permite que las ontologías sean legibles por los humanos e interpretable por las máquinas, permitiendo realizar consultas para obtener conocimiento, así como también inferir nuevo conocimiento.

El objetivo se alcanzará concretamente mediante una serie de actividades que se enuncian a continuación. Se comenzará por identificar y caracterizar las fuentes de conocimiento arquitectónico. A partir de esas fuentes se trabajará en la identificación de los conceptos, relaciones, taxonomías y se definirán las reglas que se establecen entre los conceptos y relaciones. Se explorará la posibilidad de automatizar la generación de las ontologías y su población, empleando técnicas de ontology learning (aprendizaje automático de ontologías) (Asim et al., 2018). A partir de ello, se avanzará en la propuesta de un marco de trabajo, basado en una red de ontologías, que posibilite la integración de conocimiento de arquitecturas de software (CAS) proveniente de fuentes de conocimiento heterogéneas, tanto internas como externas a la organización que desea obtener y recuperar tal conocimiento. Este marco de trabajo se aplicará en diversos escenarios de integración de fuentes de CAS. Para ello, se desarrollará una herramienta informática que posibilite la extracción de conocimiento de distintos tipos de fuentes, y la edición y ejecución de consultas sobre la red de ontologías para la recuperación de conocimiento.

El presente plan de trabajo se encuentra en desarrollo. Acualmente el trabajo de investigación está enfocado en la primera etapa que consiste en la identificación de las fuentes de conocimiento arquitectónico que poseen en las organizaciones de desarrollo de software. En la actualidad, la mayoría de las empresas de desarrollo de software han adoptado metodologías de desarrollo ágil, que suelen captar y gestionar gran parte de los procesos de desarrollo de software con algún tipo de Issue Management Systems (IMS). El seguimiento de Issues, a menudo llamado seguimiento de errores, es el proceso de seguimiento de los problemas de desarrollo abiertos en un proyecto de desarrollo de software (Henderson, 2006). Un sistema de gestión de problemas (SGI) es una aplicación informática diseñada para ayudar a garantizar la calidad del software y el apoyo a los programadores y otros interesados en el proceso de seguimiento de problemas. Por tanto, estos tipos de sistemas podrían ser una fuente de datos para realizar análisis y estudios de extracción de CAS. El término "Issue" se atribuye a la unidad de trabajo para realizar una mejora en un sistema informático. Entonces, el término "issue" puede describir la mayoría de los tipos de tareas que son necesarias para rastrear cuando se desarrolla un sistema informático (Henderson, 2006).

Luego de realizar varias búsquedas en internet se evidencia que la mayor disponibilidad de datos públicos respecto a proyectos de software se encuentra en Sistemas de Seguimiento de Incidencias registrados como issues, por ejemplo, en Jira. Como primer estudio de esta fuente de conocimiento se tomó un extracto de proyectos reales de desarrollo de software públicos en (CONNECT, 2019)(Mendeley, 2018) enfocándose en las issues que son del tipo historias de usuario (User Story). En este contexto, una historia de usuario es una breve descripción de lo que debería hacer alguna parte del software, desde la perspectiva de la persona interesada en la nueva característica que el software debería proporcionar o poseer. El uso de buenas prácticas y normas en el desarrollo de programas informáticos, como las historias de usuarios, mejora el rendimiento de la organización para hacer frente a los errores y a las solicitudes de los usuarios de nuevas

1 https://www.w3.org/RDF/

2 https://www.w3.org/TR/owl-features/

características, mejorando la organización de los equipos de desarrollo. Sin embargo, las historias de usuarios suelen estar mal escritas en la práctica y presentan defectos de calidad inherentes. Para hacer frente a esos problemas, una importante tendencia actual es la aplicación de técnicas lingüísticas computacionales a las historias de usuarios para resolver los problemas clásicos de la ingeniería de requisitos, como la formulación de requisitos de alta calidad o la creación de mejores modelos de las funcionalidades del sistema. Sin embargo, la identificación y la extracción de historias de usuarios de cientos de registros de issues no es una tarea fácil. Aunque estos sistemas ofrecen al usuario la posibilidad de categorizar explícitamente el tipo de problema, la decisión de seleccionar la categoría de un nuevo problema depende del usuario, y esta información suele omitirse o especificarse incorrectamente, lo que dificulta la posterior identificación. Como solución a esto se realizó un experimento desarrollando varios clasificadores haciendo uso de redes neuronales midiendo su precisión para clasificar issues en historias de usuario. Se desarrollaron tres modelos de redes neuronales para clasificación de textos usando TensorFlow(TensorFlow, n.d.). Se seleccionaron una red neuronal bidireccional de memoria a largo plazo y dos modelos pre-entrenados publicados por Google, ELMo(Tensorflow Hub - Google, 2018) y BERT (Tensorflow Hub - Google, 2019) para la comparación. Obteniendo que las precisiones de los tres modelos son similares, usando ELMo y la red neural bidireccional reportan una precisión del 96%, mientras que el que usa BERT muestra un 97%. Sin embargo, este último fue capaz de analizar con el mejor rendimiento el texto tanto sintáctica como semánticamente.

Siguiendo esta línea, se ha logrado crear un dataset con un total aproximado de 1.6millones de registros de distintos tipos de issues de varios proyectos que cuentan con repositorios de registro de issues abierto como Jira(IMS Jira, n.d.), RedHat(IMS Red Hat, n.d.), Apache(IMS Apache, n.d.), Sonatype(IMS Sonatype, n.d.), CONNECT(CONNECT, 2019) y Jfrog(IMS JFrog, n.d.). El dataset obtenido se encuentra actualmente en estudio con el objetivo de identificar términos relevantes en el mismo e investigar qué tipo de relaciones se puede establecer entre estos haciendo uso de las técnicas lingüísticas anteriormente mencionadas para la obtención de nuevo conocimiento. Cada uno de los registros del dataset cuenta con una columna llamada Tipo de Issue (IssueType), una descripción o resumen (Summary) y con una prioridad (Priority), en todos los casos el contenido de los registros en cada una de las columnas es texto sin estructurar expresado en lenguaje natural. Aunque se registran varios tipos de Issues los más relevantes son “Bug”, “Improvement”, “Sugestion”, “Task”, etc. como se puede observar en la Figura 1. En la Figura 2 se puede observar en una nube de palabras las palabras con mayor frecuencia en las descripciones de las issues de manera general. Mientras que en las figuras Figura 3 y Figura 4 se pueden ver las palabras mas frecuentes en las descripciones de las issues que no se clasificaron como bugs y las que si fueron clasificadas como bugs respectivamente. Si se analizan detenidamente las dos nubes de palabras se puede apreciar una tendencia distinta en las palabras comúnmente usadas en la descripción de estos distintos tipos de issue, por lo que se podría realizar algún tipo de análisis en busca de nuevo conocimiento.

IssueType	cantTypes
Bug	795932
Improvement	221087
Suggestion	128633
Task	126319
Sub-task	105188
New Feature	57732
Feature Request	47435
Enhancement	29114
Story	10751

Fig1. Tipos de Issues agrupados

Fig.2. Palabras más frecuentes en descripciones de

issues

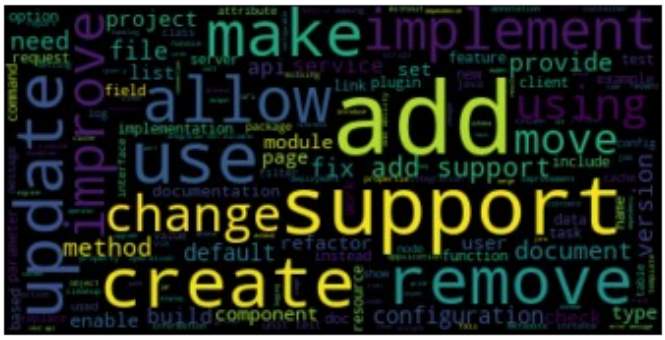

Fig3. Palabras más frecuentes en descripciones de issues de tipo no bug

Fig4. Palabras más frecuentes en descripciones de issues de tipo bug

Conclusiones

Existen varias fuentes de información, cuya representación debe ser formalizada para poder extraer de manera automática información, sin embargo, el acceso a diferentes fuentes de información en internet se dificulta debido a la escasez de repositorios públicos. Una de esas fuentes son los sistemas de seguimiento de issues los cuáles brindan una cantidad pública de datos sin estructurar comúnmente escritos en lenguaje natural. Buscando facilitar la recuperación del conocimiento que ha sido documentado o representado previamente se está llevando a cabo un estudio para tratar de identificar los requerimientos en esos registros, así como la identificación automática de los conceptos con más relevancia y las relaciones existentes entre ellos. A partir de estos dataset, se propondrán modelos para la generación automática de ontologías que caractericen algunas de las fuentes de información asociadas a los procesos de desarrollo de software.

Referencias

Akerman, A., & Tyree, J. (2006). Using ontology to support development of software architectures. *IBM Systems Journal*, *45*(4), 813–825. https://doi.org/10.1147/sj.454.0813

Asim, M. N., Wasim, M., Khan, M. U. G., Mahmood, W., & Abbasi, H. M. (2018). A survey of ontology learning techniques and applications. *Database*, *2018*(2018), 1–24. https://doi.org/10.1093/database/bay101

Capilla, R., Jansen, A., Tang, A., Avgeriou, P., & Babar, M. A. (2016). 10 years of software architecture knowledge management: Practice and future. *Journal of Systems and Software*, *116*, 191–205. https://doi.org/10.1016/j.jss.2015.08.054

Clements, P., Bachmann, F., & Bass, L. (2010). Documenting Software Architectures: Views and Beyond, Second Edition. In P. Clements (Ed.), *2002*. Addison-Wesley Professional.

CONNECT. (2019). *Navegador de incidencias - Issue Tracker CONNECT*. https://connectopensource.atlassian.net/issues/?jql=order by created DESC&startIndex=50

de Graaf, K. A., Liang, P., Tang, A., van Hage, W. R., & van Vliet, H. (2014). An exploratory study on ontology engineering for software architecture documentation. *Computers in Industry*, *65*(7), 1053–1064. https://doi.org/ https://doi.org/10.1016/j.compind.2014.04.006

Heesch, U., Avgeriou, P., & Hilliard, R. (2012). A Documentation Framework for Architecture Decisions. *The Journal of Systems & Software*, *85*, 795–820. https://doi.org/10.1016/j.jss.2011.10.017

Henderson, C. (2006). *Building Scalable Web Sites* (B. McLaughlin & S. St. Laurent (Eds.); Issue May). O'Reilly Media.

IMS Apache. (n.d.). Retrieved September 1, 2020, from https://issues.apache.org/jira/secure/BrowseProjects.jspa?selectedCategory=all&selectedProjectType=all

IMS JFrog. (n.d.). Retrieved September 1, 2020, from https://www.jfrog.com/jira/secure/BrowseProjects.jspa?selectedCategory=all&selectedProjectType=all

IMS Jira. (n.d.). Retrieved September 1, 2020, from https://jira.atlassian.com/secure/BrowseProjects.jspa?selectedCategory=all&selectedProjectType=all

IMS Red Hat. (n.d.). Retrieved September 1, 2020, from https://issues.redhat.com/secure/BrowseProjects.jspa?selectedCategory=all&selectedProjectType=all

IMS Sonatype. (n.d.). Retrieved September 1, 2020, from https://issues.sonatype.org/secure/BrowseProjects.jspa?selectedCategory=all&selectedProjectType=all

Kruchten, P., Lago, P., Vliet, H., & Wolf, T. (2005). *Building up and Exploiting Architectural Knowledge*. https://doi.org/10.1109/WICSA.2005.19

Mendeley. (2018). *Requirements data sets (user stories)*. *1*. https://doi.org/10.17632/7ZBK8ZSD8Y.1

Tang, A., Avgeriou, P., Jansen, A., Capilla, R., & Ali Babar, M. (2010). A comparative study of architecture knowledge management tools. *Journal of Systems and Software*, *83*(3), 352–370. https://doi.org/10.1016/j.jss.2009.08.032

TensorFlow. (n.d.). *TensorFlow*. https://www.tensorflow.org/

Tensorflow Hub - Google. (2018). *Elmo-Tensorflow Hub*. https://tfhub.dev/google/elmo/3

Tensorflow Hub - Google. (2019). *Bert_uncased - TensorFlow Hub*. https://tfhub.dev/google/bert_uncased_L-12_H-768_A-12/1

Uschold, M., & Gruninger, M. (1996). Ontologies: Principles, methods and applications. *Knowledge Engineering Review*, *11*(2), 93–136. https://doi.org/10.1017/s0269888900007797