Implementación en SHACL de reglas de verificación de consistencia semántica para gestión de requisitos

Luciana Tanevitch[1][0000-0002-5322-9314], Diego Torres[1,2][0000-0001-7533-0133], Leandro Antonelli[1][0000-0003-1388-0337], and Alejandro Fernández[1][0000-0002-7968-6871]

[1] LIFIA, CICPBA-Facultad de Informática, UNLP
{nombre.apellido}@lifia.info.unlp.edu.ar
[2] Departamento de Ciencia y Tecnología, UNQ

Resumen Los sistemas de gestión de proyectos permiten administrar los requerimientos que serán desarrollados. A medida que los proyectos crecen en complejidad y comienzan a intervenir más personas, es más probable que se generen inconvenientes a causa de errores en las especificaciones. La Web Semántica dispone de tecnologías para la formalización de conceptos y validación de datos que pueden aplicarse en la Ingeniería de Requerimientos para mitigar estos inconvenientes. El objetivo de este trabajo es implementar un conjunto de reglas de verificación de consistencia, completitud y calidad de requerimientos usando el lenguaje SHACL. El método propuesto es aplicado sobre un conjunto de requerimientos para mostrar la aplicabilidad y usabilidad. Este trabajo se enmarca dentro del proyecto I+D+I con alumnos "Soporte semántico para mejorar la calidad de los requerimientos" y el trabajo final de la materia Tecnologías para la Web Social Semántica, ambos desarrollados en la Facultad de Informática, Universidad Nacional de La Plata.

Keywords: Ontología de Requerimientos · Grafos de Conocimiento · Shapes

1. Introducción

El Instituto de Ingenieros Eléctricos y Electrónicos (IEEE) define un requerimiento como la capacidad o condición que un sistema debe poseer para satisfacer un documento formal de especificaciones [1]. Las herramientas de gestión de proyectos son ampliamente utilizadas en la industria del desarrollo de software ya que permiten la creación de tareas que describen requerimientos, tareas para la gestión de errores de implementación, asignación de personas encargadas de esas tareas, manejo del estado de desarrollo de una tarea (si está sin asignar, en curso o resuelta), gestión de riesgos, prioridades y costos. Sin embargo, el uso de una herramienta con estas características no garantiza la correctitud de los requerimientos, ya que los mismos pueden sufrir inconsistencias, no estar relacionados con las metas del desarrollo, o poseer problemas de calidad asociado

a riesgos innecesarios. En este sentido, la herramienta se limita a cumplir la funcionalidad que las personas ejecutan, sin verificar qué es lo que ejecutan, y esto podría conducir a un producto defectuoso.

Por su parte, la Web Semántica [3] introdujo estándares para la representación de contenido para que sea fácilmente comprensible por autómatas y así manipular y derivar información compleja de obtener para una persona. La creación y uso de taxonomías y ontologías pueden ser la base para la obtención, estructuración y gestión de la información relevante de los requerimientos [11]. Con el motivo de agilizar el desarrollo y reducir costos, el reúso de ontologías resulta una buena práctica [9]. A partir de una ontología se pueden construir bases de conocimiento representadas en un grafo de conocimiento que integre diversos conceptos. Los grafos de conocimiento describen entidades del mundo real y sus relaciones, organizándolas en forma de grafo [10] usualmente modelado en RDF para facilitar la interoperabilidad en la Web. Además, permiten derivar nuevo conocimiento a partir de la información que contienen [5].

Existen diferentes enfoques en el área de la Web Semántica que permiten representar y validar requerimientos. Diversos trabajos proponen el uso de ontologías para la conceptualización de requerimientos. En el marco de SoftWiki, un programa para la obtención, organización y gestión de requisitos, se construyó la ontología SWORE (SoftWiki Ontology for Requirements Engineering) para describir un pequeño subconjunto de los aspectos de Ingeniería de Requerimientos tales como Requerimiento y Stakeholder, pero también está alineada a FOAF y SIOC para agregar discusiones y comentarios [12], funciones muy utilizadas en las herramientas de gestión de proyectos. Antonelli et al. [2] propone el uso de una ontología para la representación de requerimientos escritos como Escenarios, y un conjunto de consultas en lenguaje SPARQL para identificar ciertos atributos de calidad de los requerimientos, como la consistencia y completitud. Siegemund et al. [14] propone un enfoque para capturar y validar la completitud y consistencia de un conjunto de requerimientos de proyectos de desarrollo usando ontologías y razonamiento basado en inferencias. En su tesis doctoral [13], detalla una ontología (Requirements Ontology) para la documentación de requisitos acorde a la especificación IEEE 830, y el conjunto de reglas basadas en axiomas para realizar verificaciones automáticas de diferentes criterios de calidad.

Si bien los lenguajes RDFS y OWL permiten definir restricciones y axiomas para realizar validaciones, éstas están limitadas a detectar inconsistencias lógicas. Shapes permite definir un conjunto de restricciones aplicables a un grafo para realizar validaciones sobre éste [7]. SHACL es un lenguaje para validar grafos de datos mediante un conjunto de condiciones representadas como shapes. La contribución de este trabajo es describir las reglas para la verificación de consistencia, calidad y completitud en requerimientos, a partir de las definidas por Siegemund et al. [13], utilizando el lenguaje SHACL, y aplicarlas sobre un grafo de conocimiento construido a partir de información obtenida de un programa de gestión de proyectos del estilo que maneja el producto Jira. Este trabajo se desarrolla en el contexto del proyecto I+D+I con alumnos "Soporte semántico

para mejorar la calidad de los requerimientos" y el trabajo final para la materia Tecnologías para la Web Social Semántica, en la Facultad de Informática de la Universidad Nacional de La Plata.

El artículo está organizado de la siguiente manera: la sección 2 introduce el uso de técnicas de Web Semántica en proyectos relacionados a requerimientos, en la sección 3 se desarrolla la contribución presentada en este trabajo y finalmente la sección 4 resume los resultados y menciona posibles trabajos futuros en relación al propuesto.

2. Contexto y trabajos relacionados

Van den Bersselaar et al. [4] proponen un prototipo para la validación de requerimientos en proyectos de construcción. En su trabajo genera un grafo con datos de un modelo de información para construcción (BIM) y un grafo de shapes a partir de los requerimientos que deberían satisfacerse. Lin et al. [8] presenta una ontología de requerimientos que da soporte a un proceso de gestión de requisitos en diferentes aspectos como trazabilidad, completitud, consistencia, satisfactibilidad, etc. Utiliza axiomas lógicos para la definición de restricciones e inferencia de nuevo conocimiento. Happel y Seedorf [6] mencionan diferentes enfoques de aplicación de las ontologías en Ingeniería de Software. En particular, destaca que es importante asegurarse que todos los participantes de un proyecto comprendan de la misma manera el problema de dominio para evitar inconvenientes en las implementaciones a causa de ambigüedad e inconsistencia, y para esto, las ontologías tienen ciertas ventajas respecto a las especificaciones semi-estructuradas debido a que automatizan las validaciones de ambigüedad y consistencia.

Existen trabajos que describen ontologías específicas del dominio en el que se posiciona esta contribución. Tappolet et al. [15] define EvoOnt, una ontología para conceptualizar la evolución del software que tiene como objetivo simplificar las tareas relativas al análisis, aplicando técnicas de la Web Semántica. SEON [16] es una familia de ontologías para el análisis semántico de la evolución del software, una de sus componentes es una ontología para representar conceptos de Jira. El vínculo de estos trabajos con el presente está dado porque las herramientas de gestión están centradas en la descripción de funcionalidades a través de *issues*, que inherentemente describen requerimientos, y conocer las relaciones entre requerimientos permitirá detectar luego los inconvenientes planteados en el dominio de estas herramientas.

3. Contribución

La contribución de este trabajo se desarrollará a lo largo de cuatro etapas que se detallan a continuación.

Definición de la ontología. Se requirió determinar los conceptos que permitirían describir los requerimientos de un proyecto. Con el objetivo de que la conceptualización sea apropiada para un dataset de un sistema de gestión de proyectos, se reusaron las ontologías SWORE y Requirements Ontology (RO),

alineando aquellos conceptos que semánticamente son equivalentes. Por ejemplo SWORE:Requirement se alineó a RO:Requirement. Se realizaron modificaciones mínimas para reemplazar las clases RO:LevelOfCost y RO:LevelOfRisk por dos propiedades análogas a las que define SWORE para *priority* y *status*.

Implementación de las reglas de verificación. En base a las reglas planteadas en el trabajo de Siegemund et al. [13], se adecuaron aquellas que resultan acordes al contexto utilizado y permitan validar la consistencia, completitud y calidad de los datos. El autor propone escoger un subconjunto de requerimientos para realizar las verificaciones llamado *configuración de requerimientos*. En este trabajo se aplican sobre la totalidad de un proyecto. Se seleccionaron y tradujeron al lenguaje SHACL las siguientes reglas. Para la característica de Completitud se escogieron las reglas: (i) Al menos una meta debe ser especificada, (ii) Cada requerimiento debe estar asociado a al menos una meta, (iii) Para cada requerimiento, se debe definir si es obligatorio u opcional. Respecto a Consistencia, se seleccionó: (iv) No debe haber requerimientos conflictivos. Y para Calidad, (v) No debe haber requerimientos opcionales con un riesgo o costo alto. A modo de ejemplo, se detalla la implementación de las reglas (i), (iv), y (v). La totalidad de la implementación puede accederse en el repositorio del proyecto [3].

```
ex:GoalCounterShape
a sh:NodeShape ;
sh:targetNode ro:Goal ;
sh:property [
  sh:path [ sh:inversePath rdf:type ] ;
  sh:minCount 1 ;
  sh:message "Al menos una meta debe ser especificada"@es;
] .
```

Regla (i) Al menos una meta debe ser especificada.

```
ex:RequirementShape
a sh:NodeShape ;
sh:targetClass swore:Requirement ;
sh:property [
  sh:path ro:isInConflictWith;
  sh:maxCount 0 ;
  sh:message "No debe haber requerimientos conflictivos"
      @es;
];
```

Regla (iv) No debe haber requerimientos conflictivos.

```
ex:RequirementShape
a sh:NodeShape ;
sh:targetClass swore:Requirement ;
sh:sparql [
   a sh:SPARQLConstraint ;
```

[3] https://github.com/tanevitch/SHACL4J

```
    sh:message "No debe haber requerimientos opcionales con
        un riesgo o costo alto"@es;
    sh:select """
        SELECT $this WHERE {
          $this a swore:Requirement .
          $this ro:isMandatory false .
          {$this sw:cost "high"} UNION {$this sw:risk "high"}
        }""";
] ;
```

Regla (v) No debe haber requerimientos opcionales con un riesgo o costo alto.

Construcción de un grafo de requerimientos. Se construyó un programa en Python que a partir de un archivo de requerimientos y la ontología definida genere un grafo RDF. Esta tarea requirió analizar cómo debían ser vinculados los datos disponibles a los conceptos que define la ontología.
Validación del grafo. A partir del grafo de datos y las reglas generadas anteriormente, el programa permite ejecutar la validación, emitiendo un informe con los inconvenientes detectados.

4. Conclusiones y trabajo futuro

En este trabajo se propone la utilización de técnicas de la Web Semántica para la verificación de requerimientos, a partir de un sistema de gestión de proyectos. El enfoque propone incorporar a estos sistemas la asistencia en la detección de problemas en la definición de los requerimientos. Jira, Trello y Pivotal Tracker son algunas de las alternativas populares que los equipos de desarrollo ágil pueden escoger para su proyecto. En general, estas herramientas permiten exportar los datos en un formato tabulado, lo que resulta práctico para procesar. Además, este artículo presenta un prototipo que permite la construcción de un grafo de conocimiento con datos obtenidos de un proyecto gestionado con el producto Jira. La propuesta de este artículo puede ser aplicada en diferentes etapas de desarrollo de un proyecto. Si el mismo se encuentra aún en desarrollo, la propuesta permite detectar en forma temprana incongruencias y minimizar los costos que produciría tratar estos errores en etapas más avanzadas. Por otra parte, si el proyecto ya ha finalizado, la validación del grafo de conocimiento permitiría detectar problemas relacionados a la calidad en la organización de los requerimientos en modo de retrospectiva general. Por ejemplo, analizar problemas en la carga de las tareas, inconsistencias cometidas o fallos en la carga de especificaciones. Queda pendiente para una futura contribución detallar el proceso de extracción de información de requerimientos escritos en lenguaje natural, y cómo se vincularían esos conceptos con las clases que define la ontología. Otra posible rama podría ser el desarrollo de una producto que pueda obtener automáticamente la información de un sistema de requerimientos, extraer la información y detectar el concepto más adecuado, definido por una ontología, con el que debe vincularse.

Referencias

1. IEEE Standard Glossary of Software Engineering Terminology. IEEE Std 610.12-1990 pp. 1–84 (1990). https://doi.org/10.1109/IEEESTD.1990.101064
2. Antonelli, L., Torres, D., Hozikian, M., Hernandez, J.E.: Semantic Support for Scenarios to Improve Communication in Agribusiness. In: Camarinha-Matos, L.M., Afsarmanesh, H., Antonelli, D. (eds.) Collaborative Networks and Digital Transformation, vol. 568, pp. 447–456. Springer International Publishing, Cham (2019). https://doi.org/10.1007/978-3-030-28464-0_38, http://link.springer.com/10.1007/978-3-030-28464-0_38
3. Berners-Lee, T., Hendler, J., Lassila, O.: The semantic web. Scientific american **284**(5), 34–43 (2001), publisher: JSTOR
4. van den Bersselaar, E., Heinen, J.J., Chaudron, M., Pauwels, P.: Automatic Validation of Technical Requirements for a BIM model using Semantic Web Technologies: 1st 4TU/14USA research day on Digitalization in the Built Environment (2022)
5. Ehrlinger, L., Wöß, W.: Towards a Definition of Knowledge Graphs
6. Happel, H.J., Seedorf, S.: Applications of ontologies in software engineering. In: Proc. of Workshop on Sematic Web Enabled Software Engineering"(SWESE) on the ISWC. pp. 5–9. Citeseer (2006)
7. Hogan, A.: The Web of Data. Springer International Publishing (2020), https://books.google.com.ar/books?id=4CPlzQEACAAJ
8. Lin, J., Fox, M.S., Bilgic, T.: A requirement ontology for engineering design. Concurrent Engineering **4**(3), 279–291 (1996)
9. Nogueira, G.G., Barcellos, M.P., Souza, V.E.S.: Towards a characterization to aid in ontology reuse. In: ONTOBRAS. pp. 138–143 (2021)
10. Paulheim, H.: Knowledge graph refinement: A survey of approaches and evaluation methods. Semantic web **8**(3), 489–508 (2017)
11. Riechert, T., Lauenroth, K., Lehmann, J.: Swore - softwiki ontology for requirements engineering. pp. 111–118 (01 2007)
12. Riechert, T., Lauenroth, K., Lehmann, J., Auer, S.: Towards semantic based requirements engineering. In: Proceedings of the 7th International Conference on Knowledge Management (I-KNOW). Citeseer (2007)
13. Siegemund, K.: Contributions to ontology-driven requirements engineering. PhD Thesis, Citeseer (2015)
14. Siegemund, K., Thomas, E.J., Zhao, Y., Pan, J., Assmann, U.: Towards ontology-driven requirements engineering. In: Workshop semantic web enabled software engineering at 10th international semantic web conference (ISWC), Bonn (2011)
15. Tappolet, J., Kiefer, C., Bernstein, A.: Semantic web enabled software analysis. Web Semantics: Science, Services and Agents on the World Wide Web **8**(2), 225–240 (Jul 2010). https://doi.org/10.1016/j.websem.2010.04.009, https://www.sciencedirect.com/science/article/pii/S1570826810000338
16. Würsch, M., Ghezzi, G., Hert, M., Reif, G., Gall, H.: SEON: A Pyramid of Ontologies for Software Evolution and its Applications. Computing **94**, 1–31 (Nov 2012). https://doi.org/10.1007/s00607-012-0204-1