Generación automática de modelos de características a partir de requisitos de dominio en lenguaje natural semiestructurado

Resumen. La ingeniería de requisitos es crucial para el éxito del desarrollo de proyectos, aunque la especificación en lenguaje natural a menudo presenta ambigüedades inherentes. Las plantillas para la representación semiestructurada de requisitos han permitido reducir la ambigüedad, mientras mantienen una escritura en pseudo-prosa fácilmente legible para los humanos. Esto se puede hacer tanto para los requisitos de sistemas individuales usando plantillas como la de Rupp y Pohl, y para requisitos de familias de sistemas, usando plantillas como SECRET (*SECurity REquirements specificaTion*). No obstante, las relaciones de variabilidad entre los requisitos quedan encapsuladas al interior de cada requisito, y para poder hacer análisis globales de esos requisitos, se necesita una visión también global de todos ellos. Obtener esta visión no es trivial y requiere de un proceso de análisis en filigrana de cada requisito para decorticar cada elemento de su estructura, y de transformación hacia un lenguaje de ingeniería que permita externalizar dichas relaciones de variabilidad. Decidimos usar los modelos de características extendidos puesto que (i) permiten representar fácilmente las relaciones de variabilidad en un dominio, (ii) las operaciones de análisis, verificación y configuración son bien conocidas, y (iii) su semántica operacional ha sido bien implementada, lo cual facilita el razonamiento automático. Este artículo propone un método para transformar requisitos escritos en SECRET hacia modelos de características. La propuesta ha sido implementada y validada mediante un caso de prueba, y la precisión de los resultados fue validada por inspección manual por los autores de este trabajo. El logro de la transformación ha puesto en evidencia la utilidad de la propuesta para facilitar el análisis a los ingenieros de requisitos.

Palabras clave: Líneas de productos de software, Modelo de características, SECRET, transformación, Ingeniería de requisitos.

1 Introducción

La ingeniería de requisitos es esencial para el éxito del desarrollo de un proyecto. Aunque se usa con frecuencia el lenguaje natural para la especificación de requisitos por la facilidad para comprender y especificar las intenciones y necesidades de las partes interesadas, a menudo se presentan ambigüedades y complejidades inherentes a la comunicación humana [1]. Las plantillas para expresar requisitos en lenguaje natural semiestructurado [2–5] han sido un paso importante hacia la solución de este problema, dado que aportan en la reducción de la ambigüedad conservando la legibilidad de los mismos. Es así como uno de los retos persistentes dentro de la ingeniería de requisitos es llevar eficazmente los requisitos expresados en lenguaje natural hacia modelos de ingeniería que posteriormente permitan tareas automatizadas de análisis y razonamiento en el

marco del desarrollo de sistemas complejos como en la Ingeniería de Líneas de Productos de Software (LPS) [6, 7]. En particular, en la Ingeniería de LPS se diseña simultáneamente un conjunto de productos o servicios similares con características tanto comunes como variables en un único proceso coordinado que maximiza la reutilización de recursos y componentes comunes entre las variantes. La LPS conjuga los modelos a nivel de dominio con sus correspondientes artefactos que permiten operacionalizarlos con el fin de crear las aplicaciones particulares que constituirán la línea de productos. Es así como la correcta especificación de requisitos para una LPS es de suma importancia para el éxito de la línea entera, puesto que en ellos no solo se especifica la necesidad de incluir o no una funcionalidad, o de respetar cierto nivel de un atributo de calidad, sino que también se expresa la variabilidad booleana de cada requisito y las eventuales relaciones de variabilidad con otros requisitos. La variabilidad booleana de un requisito se refiere a si el requisito debe hacer parte de todos los productos de la línea o solo de algunos. Que un requisito haga parte o no de ciertos productos está usualmente sujeto a restricciones que solo las pueden satisfacer otros requisitos o sus correspondientes atributos. Por ejemplo, el requisito A solo se puede incluir en un producto cuando el requisito B también haya sido incluido en el mismo producto (relación de inclusión).

Varios trabajos han demostrado la efectividad en la expresividad de modelos de ingeniería para describir un conjunto de requisitos en el área de LPS y facilitar el razonamiento sobre ellos, sin embargo, aún queda una brecha del paso de los modelos de requisitos en lenguaje natural a estos modelos de ingeniería. En esta dirección, este artículo propone un método y una herramienta para la transformación de requisitos desde una sintaxis semiestructurada hacia modelos precisos de ingeniería. Para lograr tal objetivo, se parte de la especificación de requisitos usando la plantilla SECRET [5], y se propone un método de transformación automática para obtener un modelo de características extendido. La plantilla SECRET es una guía estructural para la especificación de requisitos de dominio en lenguaje natural (inglés) semiestructurado. De la misma manera que escogimos la plantilla SECRET como modelo para la especificación de los requisitos, puesto que nos permitía especificar requisitos funcionales y no funcionales a nivel de un dominio, escogimos a los modelos de características extendidos como lenguaje de ingeniería destino puesto que (i) permiten representar fácilmente las relaciones de variabilidad más comunes en un dominio, (ii) las operaciones de análisis, verificación y configuración en dichos modelos son bien conocidas y sus implementaciones eficientes, y (iii) la semántica operacional de dicho lenguaje ha sido bien definida e implementada en herramientas computacionales, lo cual facilita el razonamiento automático sobre dichos modelos. Por lo tanto, este artículo propone un método para transformar requisitos escritos usando la plantilla SECRET hacia modelos de características. La propuesta ha sido implementada y validada mediante un caso de prueba, y la precisión de los resultados fue validada por inspección manual por los autores de este trabajo. Además, el haber podido generar un modelo de características a partir de los requisitos de dominio escritos en lenguaje natural semiestructurado ha puesto en evidencia la utilidad de la propuesta para los ingenieros de requisitos.

2 Marco conceptual

La plantilla SECRET permite especificar requisitos funcionales y no funcionales, particularmente requisitos de seguridad, para sistemas y dominios. Esta plantilla facilita un proceso guiado para especificar requisitos, distinguir entre espacios de problemas y soluciones y diferenciar los requisitos de dominio de los de aplicación. Las partes de la plantilla SECRET son: 1) Conditions - Condiciones en las que se produce un comportamiento: Algunos requisitos describen comportamientos realizados o proporcionados en condiciones específicas. 2) Family of systems, systems, or parts of a system - Familia de sistemas, sistemas o partes de un sistema: Permite especificar el nombre de la línea de productos, sistema, subsistema o componente del sistema. 3) Degree of priority - Grado de prioridad: Especifica el grado de prioridad asociado a un requisito. 4) Activity - Actividad: Especifica la caracterización de la actividad realizada por el sistema o los sistemas. 5) Security Criteria - Criterios de seguridad: Representa el objetivo u objetivos de seguridad que debe alcanzar el requisito. 6) Object or Objects - Objeto u objetos: Especifica el objeto u objetos que componen el sistema. 7) Conditionality in the object - Condicionalidad en el objeto: Describe un comportamiento que el sistema debe ejecutar si y sólo si el objeto alcanza la condición especificada. 8) Additional Object Details - Detalles adicionales del objeto. 9) Security Mechanism - Mecanismo de seguridad: Indica cómo y qué mecanismo utilizar para alcanzar los criterios de seguridad. 10) Validation Criterion or Standard - Criterio o estándar de validación: Criterio detectable para determinar el grado verificable del requisito. 11) Relax requirements statements for self-adaptive systems - Declaraciones de requisitos de los sistemas autoadaptativos: Representa la naturaleza autónoma de los requisitos en los sistemas autoadaptativos. Un ejemplo de requisito escrito con la plantilla SECRET es: “The remote monitoring system (sistema) shall (grado de prioridad) guarantee (actividad) authentication (criterio de seguridad) of a person (objeto) by role-based access control (mecanismo de seguridad)”.

Los modelos de características (Feature Models) son parte del método FODA (Feature-Oriented Domain Analysis), el cual presenta un enfoque sistemático para analizar dominios orientados en características [8]. Estos modelos permiten representar de manera intensiva muchos productos en un mismo modelo (modelo de dominio) por medio de la especificación de las características comunes, las variables y las relaciones entre ellas de tal manera que se puedan configurar y personalizar para obtener productos específicos [6] en el marco de las LPS. Al proponer el método FODA en 1990, Kang y sus colegas definieron el concepto de característica como un “aspecto visible del sistema que es relevante para algún punto de vista del sistema, o una propiedad distintiva o también puede ser una funcionalidad concreta”. Los tipos de características propuestos en FODA son: obligatorias (presentes en todos los productos), opcionales (pueden o no estar presentes), alternativas (solo una entre varias opciones) y OR (una o más entre varias opciones). Además, FODA plantea relaciones entre características de forma jerárquica (sentido de composición) y restricciones (requiere o exclusión). Así mismo, sobre los modelos de características se pueden realizar análisis automatizados de consistencia, completitud, corrección, optimización y validación del modelo. Sin embargo, a partir del método FODA se han propuesto diferentes extensiones para abordar las

necesidades de la ingeniería de LPS. Por ejemplo, se introdujo la noción de cardinalidades, atributos, referencias cruzadas, relaciones complejas (restricciones y dependencias) en el modelo de características extendido (EFM por sus siglas en inglés)[9]. Otros autores también consideraron la noción de atributos y restricciones entre atributos [10]. Los atributos se plantearon como propiedades adicionales asociadas a las características, de tipo numérico, booleano o de cadena de texto. Diversos estudios han explorado la generación de modelos de características a partir de requisitos expresados en lenguaje natural o en lenguajes estructurados, empleando una variedad de aproximaciones automáticas y técnicas avanzadas de procesamiento de lenguaje natural (PLN). Para la extracción desde requisitos expresados en lenguajes estructurados se encuentran trabajos de generación desde modelos de requisitos UML [11] centrándose en la extracción de características y la determinación de relaciones, excluyendo los atributos de las características. Para requisitos expresados en lenguaje natural, Sree-Kumar [12] aborda las limitaciones de investigaciones previas en esta área y ha desarrollado un marco de trabajo aplicando técnicas y algoritmos de aprendizaje automático para PLN. Este marco permite la extracción de características y la determinación de relaciones, enfocándose en estos aspectos y excluyendo los atributos de las características, las cardinalidades individuales y las cardinalidades grupales. Los resultados de su evaluación evidencian que es posible realizar el nombramiento de las características y la extracción de las relaciones, pero aún se presentan retos significativos para alcanzar una exactitud total.

3 Transformación de requisitos en SECRET a EFM

La generación del EFM a partir de los requisitos del sistema sigue un proceso que consta de dos pasos: primero, se especifican los requisitos utilizando la plantilla SECRET, y posteriormente, se aplican las reglas de transformación, como se ilustra en la Figura 1.

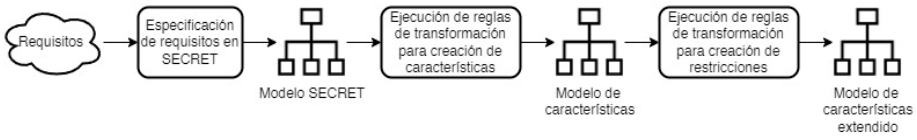

Fig. 1. Proceso de transformación

Para cada línea de productos, se construye un EFM que incluya una característica raíz con el nombre de la línea de productos. A continuación, se ejecuta una etapa para la creación de características y otra etapa para la creación de las restricciones del modelo. Esta división en dos etapas facilita el proceso de transformación debido a que es posible que los requisitos no se definan con un orden de dependencia o de asociación.

El propósito de la etapa de creación de características es identificar los requisitos que expresan funcionalidades principales y los que son refinamientos de otros. Para que un diagrama de características sea claro, es fundamental que las características tengan nombres precisos que describan la función del sistema. Por ejemplo, un requisito como "El sistema deberá permitir la autenticación de usuarios" podría traducirse en la característica "Autenticación de usuarios", mientras que "El sistema deberá enviar

notificaciones a los usuarios" podría ser "Enviar notificaciones". Encontrar las palabras correctas dentro de la especificación del requisito para el nombre correcto es complejo, debido a que pueden variar según el tipo de palabras como sustantivos o verbos en infinitivo y la posición en la que se encuentran puede ser diferente dado que la riqueza del lenguaje natural permite expresar la misma idea de diferentes formas. Los lenguajes semiestructurados como SECRET facilitan esta identificación de manera más concisa de las palabras clave, permitiendo extraer la función del sistema de la especificación del requisito para crear su correspondiente característica en el EFM. Para ello, en nuestro método todos los requisitos funcionales de la línea de productos deben someterse a las reglas de transformación delineadas en la Tabla 1 para definir el nombre correcto de la característica asociada tomando el verbo y el complemento directo del requisito.

Tabla 1. Reglas de creación de características.

Regla	Entrada (SECRET)	Salida (Feature Model)
F1	La especificación del requisito no define condiciones en la *parte Condition.*	Característica concreta con: Nombre = [<PROCESS VERB>:Activity part] + ' ' + [<Object/Asset>:Object part]
F2	La especificación del requisito define una ***Feature inclusion*** en la parte *Condition* y no define una ***Between choice*** en la *parte Object*.	Característica concreta con: Nombre = [<PROCESS VERB>:Activity part] + ' ' + [<Object/Asset>:Object part]

Una vez establecidas las características en el EFM, es necesario definir sus restricciones. La obligatoriedad de una característica puede identificarse mediante palabras clave en la especificación del requisito, como "*Todos* los sistemas de la línea de productos deben …", indicando que cada sistema debe incluir esa funcionalidad. De manera similar, la opcionalidad se reconoce con expresiones como "*Algunos* sistemas de la línea de productos deben …", sugiriendo que no todos los sistemas poseen dicha funcionalidad. Las dependencias se pueden identificar en la especificación del requisito con frases como "En caso de que la funcionalidad xyz esté *incluida*, todos los sistemas de la línea de productos deben…". Para las restricciones de cardinalidad grupal de una característica es necesario primero hallar aquellas características cuyos requisitos dependen del requisito de ésta y expresen la opción de elección al usuario de un rango de valores y no especifique una lista de posibles valores. La definición puede ser del tipo "En caso de que la característica xyz esté incluida, … el usuario podrá seleccionar entre n y m opciones" así entonces se puede crear la cardinalidad grupal, posteriormente se deben hallar las características cuyos requisitos refinen el requisito de la característica y asociarlos a la cardinalidad grupal creada. Para definir atributos en una característica es necesario hallar aquellos requisitos que dependan de ésta y expresen la opción de elección al usuario de un rango de valores pero que especifique una lista de posibles valores. La especificación es del tipo "En caso de que la característica xyz esté incluida, … el usuario podrá seleccionar entre n y m opciones (a, ..., z)" así entonces se puede crear el atributo opciones en la característica xyz con esos posibles valores. Para este propósito, todos los requisitos funcionales de la línea de productos deben someterse a las reglas de transformación delineadas en la Tabla 2.

Tabla 2. Reglas de creación de las restricciones.

Regla	Entrada (SECRET)	Salida (EFM)
C1	La especificación del requisito no define condiciones en la parte *Condition* y define ***All systems*** en la parte *System*.	Relación obligatoria desde la característica Root hasta la característica concreta asociada con el requisito.
C2	La especificación del requisito no define condiciones en la parte *Condition* y define ***Some systems*** en la parte *System*.	Relación opcional desde la característica Root hasta la característica concreta asociada con el requisito.
C3	La especificación del requisito define una ***Feature inclusion*** en la parte *Condition* y define ***All systems*** en la parte *System* y no existe una relación *Refinement* desde el requisito hasta el requisito asociado a la característica en ***Feature inclusion***	Relación obligatoria desde la característica concreta asociada a [*<Included feature>:Feature inclusion*] hasta la característica concreta asociada al requisito.
C4	La especificación del requisito define una ***Feature inclusion*** en la parte *Condition* y define ***Some systems*** en la parte *System* y no existe una relación *Refinement* desde el requisito hasta el requisito asociado a la característica en ***Feature inclusion***	Relación opcional desde la característica concreta asociada a [*<Included feature>:Feature inclusion*] hasta la característica concreta asociada al requisito.
C5	La especificación del requisito define una ***Feature inclusion*** en la parte *Condition* y define una ***Between choice*** en la parte *Object* y define una lista en ***Additional object details*** en la parte *Object details* y existe una relación Refinement desde el requisito hasta el requisito asociado a la característica en ***Feature inclusion***	Atributo en la característica asociada a [*<Included feature>:Feature inclusion*] con: Nombre = [*<Object/Asset>:Between choice*] Dominio = String value Valores posibles = [*<Additional object details>:Additional object details*] Restricción = '[' + [*<A>:Between choice*] + '..' + [*<B>:Between choice*] + ']'
C6	La especificación del requisito define una ***Feature inclusion*** en la parte *Condition* y define una ***Between choice*** en la parte *Object* y no define una lista en ***Additional object details*** en la parte *Object details* y existe una relación Refinement desde el requisito hasta el requisito asociado a la característica en ***Feature inclusion***	Cardinalidad grupal desde la característica concreta asociada a [*<Included feature>:Feature inclusion*] con cardinalidad: m = [*<A>:Between choice*] n = [*<B>:Between choice*]
C7	La especificación del requisito define una ***Feature inclusion*** en la parte *Condition* y no define una ***Between choice*** y existe una relación *Refinement* desde el requisito hasta el requisito asociado a la característica en ***Feature inclusion***	Relación desde la cardinalidad grupal asociada con la característica concreta [*<Included feature>:Feature inclusion*] hasta la característica concreta asociada con el requisito.
C8	Existe una relación *Conflicting* desde el requisito hasta otro requisito funcional.	Relación de exclusión desde la característica concreta asociada al requisito hasta la

		característica concreta asociada al otro requisito con el que entra en conflicto.
C9	Existe una relación *Dependency* desde el requisito hasta otro requisito funcional.	Relación de inclusión desde la característica concreta asociada al requisito hasta la característica concreta asociada al otro requisito con el que hay dependencia.

4 Implementación de la solución

La funcionalidad de transformación automatizada de modelos SECRET a EFM la desarrollamos en la herramienta VariaMos, la cual permite construir LPS bajo un enfoque de ingeniería dirigida por modelos. VariaMos ofrece el lenguaje de modelado Domain Requirements AC para la especificación de requisitos de LPS en formato SECRET y el lenguaje Feature Model with attributes para crear EFM. Nuestra aproximación consistió en extraer los elementos necesarios de los requisitos modelados en este lenguaje y aplicar las reglas de transformación al EFM. Para ello, fue necesario mapear cada palabra de la especificación del requisito a la parte correspondiente de la plantilla SECRET. Creamos un lenguaje para modelar grafos extendidos que representara las posibles secuencias de palabras de la plantilla SECRET, con cada nodo correspondiente a una palabra y con una etiqueta indicando a qué parte de la plantilla pertenecía. Para elementos de SECRET que permiten múltiples palabras, por ejemplo, el nombre de la línea de productos, utilizamos nodos especiales. Este grafo nos permitió, a nivel de código, generar un diccionario para el mapeo en el algoritmo de transformación. Dado que VariaMos es un framework en la nube basado en microservicios, creamos un servicio en Node.js y lo integramos al ecosistema de la herramienta. Este microservicio recibe como petición el modelo SECRET en formato JSON diseñado por el usuario, luego el algoritmo extrae la especificación de cada requisito y, utilizando el diccionario de mapeo, busca el camino del grafo que se ajusta al texto. Una vez identificados los nodos del camino seleccionado, se determina la parte correspondiente de la plantilla SECRET a partir de la etiqueta de cada nodo y se aplican las reglas de transformación para crear el EFM, que luego se devuelve en formato JSON a la aplicación principal que lo interpreta gráficamente para que el usuario lo visualice.

5 Validación preliminar con una prueba de concepto

Consideremos una línea de productos de software para editores de texto. En esta línea, se formulan los requisitos funcionales para la edición, formato y almacenamiento de documentos, utilizando el formato SECRET dentro del modelo de requisitos del lenguaje Domain Requirements AC de VariaMos. Algunos de los requisitos son: FR14: Some systems of the text editor product line could provide user with the ability to create the lists.; FR15: In case create lists is included all systems of the text editor product line shall provide user with the ability to create the numeric lists; y FR16: In case create numeric lists is included all systems of the text editors product line shall provide user with the ability to choose between 1 and 2 types (roman, arabic). Una vez definido el

modelo de requisitos, se procede a la transformación automatizada en un EFM con la herramienta, como se detalla en la Figura 4. Al completar este proceso de transformación, se obtiene el EFM que se presenta en la Figura 5[1].

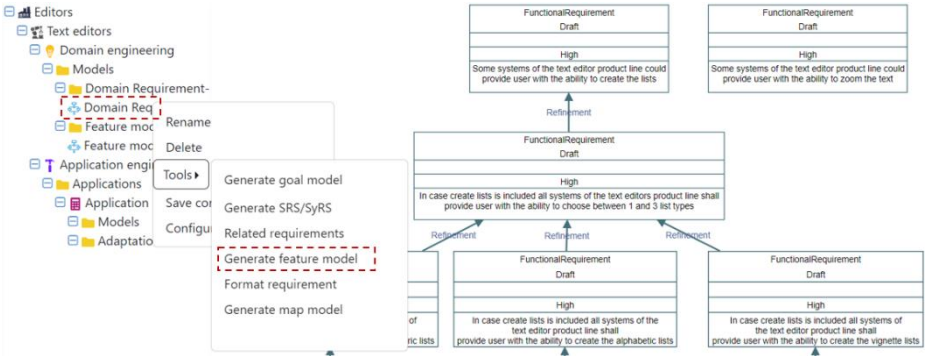

Fig. 4. Transformación al modelo de características extendido.

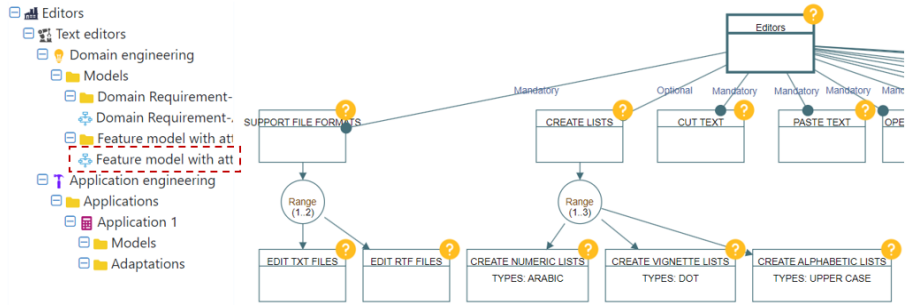

Fig. 5. Modelo de características extendido generado.

6 Conclusiones y trabajo futuro

La ingeniería de requisitos es esencial para el éxito de los proyectos, aunque la especificación en lenguaje natural presenta ambigüedades. Las plantillas semiestructuradas han reducido estas ambigüedades, manteniendo una escritura legible. Esto aplica a requisitos de sistemas individuales y familias de sistemas. Sin embargo, las relaciones de variabilidad entre los requisitos de dominio suelen quedar encapsuladas, dificultando su análisis global y requiriendo un minucioso proceso de análisis y transformación hacia un lenguaje de ingeniería. Los modelos de características extendidos permiten representar relaciones de variabilidad comunes, con operaciones de análisis y configuración bien conocidas y eficientes, y su semántica operacional está bien definida en herramientas computacionales. Este artículo propone un método para transformar requisitos escritos con la plantilla SECRET hacia EFM; la propuesta se ha implementado y

[1] Los recursos completos están disponibles en https://github.com/variamosple/examples/tree/main/text_editors

validado con un caso de prueba, y la precisión de los resultados fue verificada manualmente. Generar un modelo de características a partir de requisitos de dominio en lenguaje natural semiestructurado ha demostrado ser útil para los ingenieros de requisitos, permitiéndoles identificar contradicciones, determinar la viabilidad de requisitos y verificar configuraciones. Como trabajos futuros, se identifican transformaciones a otros modelos como KAOS y la integración de metodologías como Secure Tropos.

Referencias

1. Berry, D., Kamslies, E.: The syntactically dangerous all and plural in specifications | IEEE Journals & Magazine | IEEE Xplore, https://ieeexplore.ieee.org/abstract/document/1377124
2. Rupp, C., Joppich, R.: Anforderungsschablonen. En: Requirements-Engineering und-Management. pp. 215-246. Carl Hanser Verlag München (2014)
3. Mazo, R., Jaramillo, C.A., Vallejo, P., Medina, J.H.: Towards a new template for the specification of requirements in semi-structured natural language. Journal of Software Engineering Research and Development. 8, 3 (2020). https://doi.org/10.5753/jserd.2020.473
4. Mavin, A., Wilkinson, P., Harwood, A., Novak, M.: Easy approach to requirements syntax (EARS). Presentado en octubre 4 (2009)
5. Hnaini, H., Mazo, R., Vallejo, P., Lopez, A., Champeau, J., Galindo, J.: SECRET: A New SECurity REquirements SpecificaTion Template. En: Rocha, Á., Ferrás, C., Hochstetter Diez, J., y Diéguez Rebolledo, M. (eds.) Information Technology and Systems. pp. 235-246. Springer Nature Switzerland, Cham (2024)
6. Pohl, K., Böckle, G., Van Der Linden, F.: Software Product Line Engineering. Springer, Berlin, Heidelberg (2005)
7. Mazo, R. (ed): Guía para la adopción industrial de líneas de productos de software. (2018)
8. Kang, K.: Feature-Oriented Domain Analysis (FODA) Feasibility Study, https://insights.sei.cmu.edu/library/feature-oriented-domain-analysis-foda-feasibility-study/
9. Czarnecki, K., Grünbacher, P., Rabiser, R., Schmid, K., Wąsowski, A.: Cool features and tough decisions: a comparison of variability modeling approaches. En: Proceedings of the Sixth International Workshop on Variability Modeling of Software-Intensive Systems. pp. 173-182. ACM, Leipzig Germany (2012)
10. Benavides, D., Trinidad, P., Ruiz-Cortés, A.: Automated Reasoning on Feature Models. En: Pastor, O. y Falcão e Cunha, J. (eds.) Advanced Information Systems Engineering. pp. 491-503. Springer, Berlin, Heidelberg (2005)
11. Casalánguida, H., Durán, J.E.: Automatic generation of feature models from UML requirement models. En: Proceedings of the 16th International Software Product Line Conference - Volume 2. pp. 10-17. Association for Computing Machinery, New York, NY, USA (2012)
12. Sree-Kumar, A., Planas, E., Clarisó, R.: Extracting software product line feature models from natural language specifications. En: Proceedings of the 22nd International Systems and Software Product Line Conference - Volume 1. pp. 43-53. Association for Computing Machinery, New York, NY, USA (2018)