Hacia una Meta-Herramienta de Análisis Automático de Modelos de Variabilidad

Esteban Ruiz de Galarreta *1, Matias Pol'la*12
Agustina Buccella*12 , Alejandra Cechich*1
1- *GIISCo, Facultad de Informática - Uncoma*
2- *CONICET, Consejo Nacional de Investigaciones Científicas y Técnicas*

Universidad Nacional del Comahue, Neuquén, Argentina
esteban.ruizdegalarreta@fi.uncoma.edu.ar, matias.polla@fi.uncoma.edu.ar,
agustina.buccella@fi.uncoma.edu.ar, alejandra.cechich@fi.uncoma.edu.ar

Abstract. El análisis automático de modelos de variabilidad es una actividad clave dentro de la gestión de la variabilidad. Existe un variado número de investigaciones y de enfoques que se han centrado en esta temática, fundamentalmente debido a que el hecho de determinar la validez de los modelos e identificar los problemas que contiene tanto a nivel de la definición de una línea de productos de software así como la instanciación y generación de productos en etapas tempranas, agiliza el desarrollo de la misma. Estos numerosos enfoques presentan diferentes tipos de modelos con distintas reglas, elementos y dependencias; diferentes procesos de validación y diversas reglas lógicas que soportan el análisis, así como diferentes solver (resolvedores lógicos) que determinan la validez de los modelos. Teniendo en cuenta este contexto, en este trabajo presentamos una Meta-herramienta de análisis automático basada en un proceso general, con el objetivo de proveer la suficiente flexibilidad que permita ser adaptada a diferentes modelos, reglas y resolvedores lógicos.

1 Introducción

La gestión de la variabilidad es una actividad dedicada a proporcionar flexibilidad y un alto nivel de reutilización durante el desarrollo del software. Dentro del enfoque de Líneas de Productos de Software (LPS), las actividades relacionadas con la gestión de variabilidad son las encargadas de proveer la flexibilidad necesaria para desarrollar un conjunto de aplicaciones similares basadas en un rango manejable de funcionalidades variables según las necesidades de los usuarios expertos. Dentro de la gestión de variabilidad, ha surgido un nuevo campo de investigación y estudio denominado análisis (automatizado) de variabilidad, centrado específicamente en la validación de modelos de variabilidad de acuerdo con un conjunto de problemas o errores predefinidos [2, 6, 10, 13]. Aunque la comunidad de investigación ha llegado a algún consenso sobre los escenarios base que deben evaluarse, el gran número de enfoques de modelado hace que la forma de evaluar estos escenarios sea investigada extensamente en la actualidad. La comprensión de los enfoques de modelado es fundamental para analizar las actividades de validación aplicadas a estos modelos. Por lo tanto, las actividades

de validación o el análisis de variabilidad deben considerarse desde estas perspectivas de modelado. Por ejemplo, el proceso de análisis debe incluir un conjunto diferente de tareas cuando la especificación se basa en un solo modelo [5] o cuando se representan diferentes modelos enlazados [3, 11]. Además, es importante considerar el lenguaje formal utilizado para traducir estos modelos de variabilidad porque éste es el punto de partida para realizar un análisis automatizado [2]. En la búsqueda de la automatización del proceso de verificación y validación de modelos de variabilidad, se han desarrollado diferentes herramienta que abordan la temática teniendo en cuenta características específicas de modelos de variabilidad particulares, como por ejemplo S.P.L.O.T. [9] y Variamos [8] entre otras. Teniendo en cuenta este contexto, en este trabajo se propone una meta-herramienta de análisis automatizado, basado en un proceso general de verificación presentado en [2]. Esta herramienta fue desarrollada con la flexibilidad necesaria para adaptarse a modelos de variabilidad, lógicas asociadas a los modelos y reglas particulares a las necesidades de cada usuario potencial.

Este trabajo se organiza de la siguiente manera: en la siguiente sección se detalla el marco teórico y trabajos relacionados. En la sección 4 se describe la meta-herramienta junto con su arquitectura, y el modelo de variabilidad asociado que permite su adaptación. En la sección 5 se detalla un caso de estudio en donde se instancia la meta-herramienta con nuestro propio modelo de variabilidad, reglas lógicas y escenarios de validación. Por último se detallan las conclusiones y trabajos futuros.

2 Marco Teórico y Trabajos Relacionados

El objetivo principal de una LPS es proveer una plataforma común lo suficientemente flexible de manera que permita adaptarse a las diferentes necesidades de los diversos productos dentro del rango de requerimientos establecido. Dicha flexibilidad se logra mediante la identificación, definición y posterior configuración de la variabilidad. La misma se define en [1] como la *capacidad de un artefacto de software de ser configurado, adaptado, extendido o cambiado en un contexto específico*.

La gestión de variabilidad [11] involucra todas las actividades relacionadas con la variabilidad a lo largo de todo el ciclo de vida de una LPS, desde la especificación de requerimientos, modelado e implementación, hasta el testing de los productos derivados. Entre las mismas podemos incluir a la definición de la variabilidad junto con los puntos de variación, la gestión de los componentes variables y la resolución de la misma al momento de realizar el proceso de derivación.

Si nos centramos en análisis automático de la variabilidad, existen una gran variedad de trabajos y propuestas que abordan dicho tema, centrándose en actividades de validación específicas para la fase de ingeniería de dominio del desarrollo de LPS. Algunos enfoques que presentan surveys o revisiones se pueden encontrar en [2, 15]. Por ejemplo, en [15] los autores presentan una comparación de análisis automatizado usando Alloy [4] para modelos de características (Feature Model - FM). En [2] existe una revisión bibliográfica más amplia del análisis automático de FM, basado en un proceso general que define un conjunto de tareas para evaluar FMs. En la Figura 1 podemos ver una adaptación de este proceso, en la cual identificamos cinco componentes principales,

definiendo un proceso general que sienta las bases de la actividad de la verificación automática de modelos.

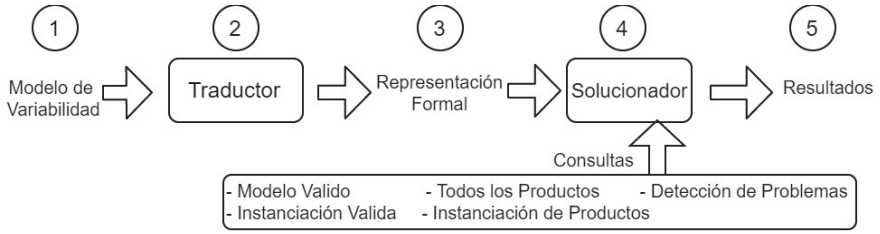

Figura 1. Proceso general para analisis automatico de modelos de variabilidad

El primer componente es el *modelo de variabilidad* que puede definirse siguiendo cualquiera de los diferentes enfoques de modelado. El segundo componente es el *traductor*, que realiza el proceso de transformación entre el modelo y la representación formal teniendo en cuenta el enfoque seleccionado. El tercer componente es el *modelo formal* o representación formal, que incluye la variabilidad en términos lógicos. El cuarto componente es un *solver*, responsable de validar el modelo formal. Este componente también recibe el conjunto de escenarios o consultas de validación, que determinan los *resultados* (quinto componente) del proceso de análisis.

Centrándonos en el modelo de variabilidad, podemos encontrar un gran número de trabajos proponiendo diferentes enfoques para modelar la variabilidad. Entre ellos se destacan tres propuestas altamente referenciadas y utilizadas en el desarrollo de LPS: Modelo de características (FM)[5], Modelos de Variabilidad Ortogonal (Orthogonal Variability Model - OVM) [11] y Lenguaje de Variabilidad Común (Common Variability Language - CVL) [3] [1]. Si bien la expresividad de cada modelo propuesto podría ser similar, existen diferencias sustanciales entre ellos. Las propuestas basadas en FM presentan un modelo jerárquico que permite describir los puntos variantes en común y las variabilidades dentro de una misma estructura. OVM permite especificar variabilidades en modelos separados pero vinculados (por medio de enlaces); y CVL es un lenguaje independiente del dominio para especificar la variabilidad, en el que los modelos se expresan en DSL (Domain Specific Language) o UML.

En cuanto a los enfoques que presentan soluciones para un proceso de análisis automatizado, durante los últimos años han surgido nuevos enfoques en este campo proponiendo novedosas ideas [7, 9, 13, 14]. Un resumen de algunas de estas propuestas se muestra en la Tabla 1 de acuerdo con cuatro aspectos: la herramienta soporte (S.P), el modelo de variabilidad (V.M.), el modelo formal (M.F.) y el solver. La columna de herramienta de soporte se establece como P cuando la herramienta es un prototipo, T cuando se trata de un complemento o herramienta de escritorio o WT cuando se trata de una herramienta web.

[1] Cada categoría contiene un gran número de propuestas, que abarcan diferentes aspectos

Enfoque	S.P.	V.M.	F.M.	Solover
Lauenroth et al. [7]	P	OVM I-O/automata	CTL	SAT-VM
FAMA [14]	T	OVM -FM	CSP	BDD SAT Choco
Metzger et al. [10]	P	OVM - FD	CNF	SAT
VariaMos [8]	T	Independiente	CSP	SWI-Prolog
S.P.L.O.T. [9]	WT	FM	3-CNF	SAT

Tabla 1. Resumen de Enfoques de Analisis Automatico de Modelos de Variabilidad

Finalmente, otro aspecto importante a analizar es el conjunto de consultas que los solvers son capaces de responder. En este sentido, en la literatura existen varias definiciones del conjunto de anomalías o desajustes que pueden encontrarse en un modelo de variabilidad [2, 6, 16]. En estos trabajos se destacan consultas como modelo válido, instanciación valida y características muertas (Dead Feature).

3 Antecedentes

En trabajos anteriores [12] hemos definido un modelo de variabilidad propio, llamado SeVaTax, que toma como entrada modelos de variabilidad basados en primitivas OVM. En este trabajo, presentamos una meta-herramienta cuyo objetivo es poder ser adaptada a las diferentes características y particularidades que pueden presentar los diversos modelos de variabilidad. Como caso de estudio, tomaremos nuestro modelo SeVaTax para analizar la adaptación de la misma.

En términos generales el modelo SeVaTax está basado en OVM, que permite ser diseñado por medio de un conjunto de modelos interrelacionados llamados Datasheets; y está compuesto por un conjunto de puntos variables, dependencias y operadores de alcance. Dentro de los puntos de variabilidad utilizamos ObligatorioVP, OpcionalVP, AlternativoVP (una opción de un conjunto) y VarianteVP (al menos una opción de un conjunto). En cuando a las dependencias desarrollamos tres dependencias: usa (requiere mutuo), requiere y excluye. Por último presentamos dos operadores de alcance, que definen el alcance de cada uno de los puntos variantes (tipos de variabilidad) presentes en el modelo. Punto variante global determina que el punto variante sólo tendrá una sóla configuración a en todo el modelo; y punto variante específico permite que el punto variante contenga diferentes configuraciones en diferentes casos. Además para nuestro modelo hemos definido una lógica que rige la validación automática basada en la lógica proposicional representada en CNF (Forma normal conjuntiva).

4 Meta-Herramienta de validación autumatica de Modelos de Variabilidad

La meta-herramienta, está basada en el metamodelo de la Figura 2 (A) que especifica los diferentes aspectos y características de cada uno de los elementos que conformarán

la instancia de la herramienta y que permitirán la posterior evaluación. Entre estos aspectos y características es necesario definir principalmente el modelo de variabilidad que será analizado junto con sus reglas de traducción.

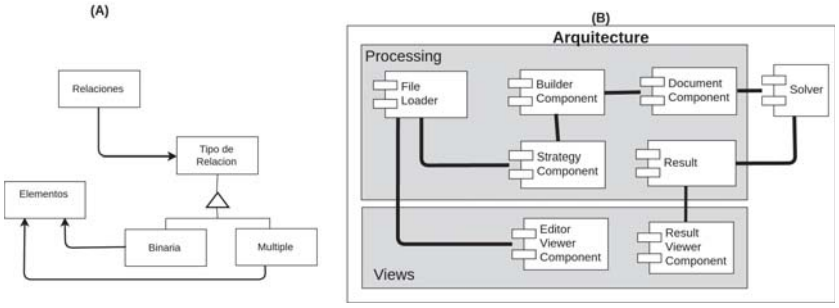

Figura 2. (A)Modelo estructural de la Meta-herramienta - (B) Arquitectura de la Meta-herramienta

En la Figura 2 (A) podemos visualizar los conceptos más básicos del meta-modelo utilizado para la definición de nuestra herramienta que fueron mencionados anteriormente. Estos conceptos son los de relaciones, su tipo y los elementos que componen a las mismas. Dichos tipos de relaciones, nos permiten que las mismas sean de dos tipos diferentes, uno es el tipo binario, que como bien dice su nombre, nos limita a una relación entre dos elementos de nuestra estructura. Por otro lado, existe la múltiple, que es la utilizada cuando se necesita relacionar más de dos elementos. Una vez entendidos estos conceptos básicos, se debería ser capaz de lograr una definición de un modelo requerido para el funcionamiento de la herramienta. Dicho diagrama, puede ser extendido a futuro para ampliar las características disponibles en el meta-modelo de nuestra herramienta,

En cuanto al modelo de variabilidad es necesario definir los elementos que lo conforman, las relaciones o dependencias pueden tener dichos elementos entre ellos (obligatorio, opcional, alternativa, variante, entre otras); y cuáles son sus posibles restricciones (uso, de requerimiento y de exclusión). De esta manera estaremos definiendo aspectos sintacticos y semanticos del modelo. Además es necesario definir cómo se debe comportar nuestra herramienta ante cada una de estas situaciones al momento del análisis de correctitud. Por lo tanto, se debe seleccionar y definir una traducción lógica de cada elemento, relación y restricción.

4.1 Proceso general

En esta sección explicaremos en detalle la arquitectura de la herramienta presentada en este trabajo. Como se mencionó anteriormente el objetivo de esta herramienta es presentar una estructura que se adapte a las particularidades de diversos modelos de variabilidad con diferentes elementos, relaciones y reglas para determinar la validez o no de cada modelo.

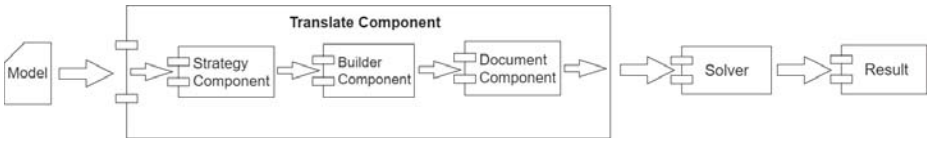

Figura 3. Proceso de analisis automatico de la Meta-herramienta

La Figura 3 muestra un proceso general implementado en este trabajo. Este proceso debe ser lo suficientemente flexible como para adaptarse a diferentes configuraciones de distintos modelos de variabilidad. Para ello hemos tomado el proceso general de verificación automático propuesto en [2] como base para armar la meta-herramienta de validación. Como puede observarse en la Figura, este proceso está compuesto por cinco componentes y siete flujos de entrada/salida. Como se puede observar, la entrada principal es un modelo de variabilidad (conjunto de hojas de datos funcionales llamados datasheets particularmente para nuestro caso de estudio), que inician el proceso. Este modelo o conjunto de modelos es tomado por el "Traductor", que está formado por 3 sub-componentes de software, que detallaremos a continuación:

- El primero de ellos es el componente 'Strategy' que se encarga de implementar el procedimiento que marcará las reglas utilizadas para traducir las entradas que posee un datasheet. Este, para cada punto variante de cada servicio, estipula el formato que se añadirá al archivo de resultado final.
- El componente 'Builder' recibe como entrada la estructura de las reglas dispuestas en el componente 'Strategy' y se encarga de representarlas en un documento.
- El componente 'Document' se encarga de diagramar el formato del resultado final de todo el proceso, estipulando la disposición final de cada regla generada con anterioridad en el componente 'builder'. Esta disposición, está sumamente relacionada con lo que un componente 'solver' pueda necesitar.
- El componente 'Solver' es el encargado de que luego de resolver lógicamente un documento que contiene todas las traducciones de los modelos de variabilidad requeridos provisto por las etapas anteriores del proceso. Este componente, es el que nos marcara si dicha traduccion es soluble o no, y si no lo es, qué reglas (y por lo tanto, sus servicios) están causando tal error.
- Por último, el componente 'Result' es el encargado de interpretar y darle un formato más amigable a la salida provista por el componente 'solver', para así simplificar datos redundantes o que normalmente no son necesarios en ciertos casos.

Para visualizar la flexibilidad buscada con esta meta-herramienta, hemos desarrollado un diagrama de variabilidad, diseñado a través de nuestras reglas de modelado (sección 3). En este sentido el diagrama de variabilidad para la arquitectura de la meta-herramienta, que determina cómo se pueden variar los componentes utilizados en cada parte del proceso. Dicho diagrama se puede observar en la Figura 4, donde se puede apreciar la variabilidad de cada componente explicado anteriormente. El diagrama de variabilidad está compuesto por 6 puntos variantes y 3 tipos de variabilidad (Obligatorio, Alternativo y Variante). Por ejemplo, como podemos observar, el punto variante

especificado para el componente del Solver, especifica que existe un punto variante entre un servicio llamado DL-Solver y otro denominado SAT-Solver. Así mismo, el servicio SAT-Solver tiene su propio punto variante, en este caso, es un punto variante alternativo, que nos indica que debemos instanciar dicho servicio con "Sat4J" o bien con "Choco".

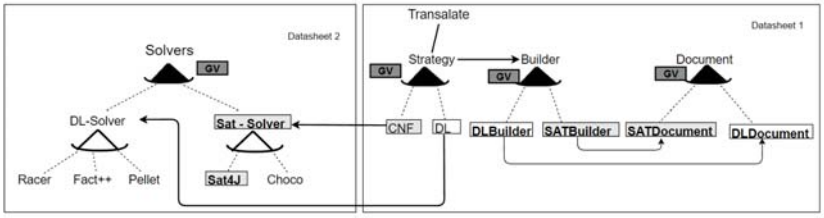

Figura 4. Modelo de Variabilidad de la Meta-Herramienta

Por medio de los servicios disponibles en el modelo de variabilidad podemos apreciar la existencia de implementaciones de distintas características para cada servicio, como por ejemplo, el componente/servicio de strategy, posee un punto variante global con los servicios de CNF (Forma normal conjuntiva) y DL (Lógica Descriptiva), que nos indica que estas son algunas de las dos posibles estrategias disponibles para aplicar a la hora de traducir una hoja de datos funcional. Además, dicho servicio, tiene una dependencia de uso con el servicio denominado "Builder". Esta relación lo que determina es que, cualquier servicio elegido del punto global de "Strategy", necesitará para su correcto funcionamiento, poder comunicarse con una de las tantas posibles instanciaciones del componente "Builder". Así mismo, el modelo de variabilidad también nos indica que el servicio denominado "SATBuilder" que pertenece a la categoría de los "Builders", también tiene una relación de uso con su homónimo de la categoría "Document".

En nuestro caso, para la realización de la implementación propuesta, se optó solamente por implementar aquellos servicios que se encuentran coloreados con un gris de fondo. El resto de las implementaciones posibles, quedan como futuros trabajos a realizar.

4.2 Arquitectura de la Meta-herramienta

La Figura 2 (B) detalla la arquitectura de la meta-herramienta de validación, la cual está basada en una arquitectura genérica de cliente-servidor. Como podemos ver, la arquitectura se divide en 3 capas, la primera es determinar los componentes utilizados en la etapa de procesamiento, es decir, los componentes que se encontrarán del lado del servidor. Por otra parte, se visualizan los componentes que se encargará de mostrarle al usuario los resultados obtenidos en todo el proceso, estos, fueron demarcados en la zona de Views, tales como el componente encargado de ofrecernos una visual gráfica para la herramienta (Editor Viewer Component) y el ya mencionado componente que

nos ayuda a visualizar los resultados obtenidos del componente ‘solver’ (Result Viewer Component), donde los mismos, en nuestra implementación se alojarán del lado del cliente. Por último, en la arquitectura también utiliza un componente externo a nuestra implementación como es el caso del solucionador (solver).

5 Caso de estudio: Instanciación herramienta de validación

En esta sección explicaremos el proceso de instanciación realizado para adaptar nuestro proceso SeVaTax (introducido en la Sección 3) a la meta-herramienta. En la siguiente subsecciones explicaremos en detalle cada uno de los elementos instanciados.

5.1 Modelo SeVaTax

Para adaptar el modelo SeVaTax hemos instanciado el meta-modelo. Éste quedó compuesto por servicios de software como los elementos del modelo, relacionados por medio de los tipos de variabilidad de SeVaTax 3. Cabe destacar que la instanciación del meta-modelo puede ser definida en cualquier lenguaje de texto estructurado que brinde la capacidad para definir jerarquías de elementos. Nuestra instanciación fue definida utilizando la estructura provista por el formato de texto JSON (JavaScript Object Notation) para evitar futuros conflictos con la herramienta implementada. Una vez que se definió el modelo , definimos un conjunto de elementos de prueba o casos de test, que fueron la guía utilizada para el desarrollo de la instanciación de la herramienta.

5.2 Herramienta de Analísis Automatizado

Para el desarrollo de la instanciación de la herramienta, se utilizó el framework de Typescript "Angular 2". Teniendo en cuenta la tecnología seleccionada, hemos implementado cada uno de los componentes de arquitectura detallados en la figura 2 (B). Para esto, primero se implementó el componente "Editor Viewer Component" para que el usuario logre editar y brindar al servidor los datos necesarios para la traducción. El componente de "File Loader", fue implementado de manera de descomponer los documentos JSON mediante la búsqueda de los diferentes puntos variantes de los diferentes servicios. Para cada uno de estos puntos variantes, el componente identifica la información de los mismos y la prepara para el componente "Strategy". Este último realiza el análisis de cada elemento perteneciente al modelo definido, reconociendo entre otras cosas el tipo de variabilidad (Mandatario, opcional, alternativo o variante). Luego, procede a traducirlo aplicando las reglas de traducción especificados en la estrategia requerida. Para nuestro experimento, se utilizó la estrategia CNF propias del modelo SeVaTax. Finalmente, se implementó el "Builder", que se encarga de recibir los resultados y transformarlos de manera tal que se cumpla lo estipulado por el componente "Document". Este último componente fue implementado de manera de estructurar un documento con el formato específico de CNF para SAT4J.

5.3 Validación y Experimentación

Una vez implementada la instanciación de la herramienta, se desarrollaron diferentes casos de prueba para evaluar individualmente cada una de las dependencias del modelo utilizado, junto con un caso de prueba que combina todas las dependencias y relaciones del modelo. Cada uno de estos casos de test fueron ejecutados en la instanciación de la herramienta, y otorgaron resultados positivos en cuanto a la correctitud de la solucion, que fue el único criterio evaluado en esta instancia. Por lo tanto, podemos remarcar que la Meta-Herramienta implementada se adaptó correctamente al modelo SeVaTax sin mayores complicaciones. Esto, nos brinda un panorama alentador con respecto a posibles mejoras en la misma, ya que la totalidad de los experimentos efectuados con la herramienta, entregaron los resultados correctos, lo que nos permitirá adicionar mayor complejidad y robustez al validador, a través de la utilización de diferentes modelos de variabilidad, validadores y reglas de análisis.

6 Conclusiones

En este trabajo hemos presentado una Meta-herramienta de análisis automático de variabilidad la cual nos permite adaptar diferentes modelos y reglas, generando diversas instanciaciones acorde a diferentes enfoques y particularidades. En el mismo se demuestra la capacidad de adaptación de la misma, por medio de un caso de estudio basado en el modelo de variabilidad SeVaTax desarrollado anteriormente.

Como trabajo futuro, se buscará ampliar la gama de modelos de variabilidad instanciados por la herramienta, de forma de darle una mayor flexibilidad y adaptación a los diferentes enfoques. Además estamos trabajando en la traducción a lógicas DL de nuestro modelo de variabilidad SeVaTax. Esto nos abrirá la posibilidad de configurar diferentes solvers y evaluar la adaptabilidad de la meta-herramienta a este conjunto de características.

Bibliografía

[1] Felix Bachmann and Paul Clements. Variability in software product lines. Technical Report CMU/SEI-2005-TR-012, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 2005.

[2] David Benavides, Sergio Segura, and Antonio Ruiz-Cortés. Automated analysis of feature models 20 years later: A literature review. *Inf. Syst.*, 35(6):615–636, September 2010.

[3] Ø. Haugen, B. Møller-Pedersen, J. Oldevik, G. K. Olsen, and A. Svendsen. Adding standardized variability to domain specific languages. In *2008 12th International Software Product Line Conference*, pages 139–148, Sept 2008.

[4] Daniel Jackson. *Software Abstractions: Logic, Language, and Analysis*. The MIT Press, 2006.

[5] K. Kang, S. Cohen, J. Hess, W. Nowak, and S. Peterson. Feature-Oriented Domain Analysis (FODA) Feasibility Study. Technical Report CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University Pittsburgh, PA., 1990.

[6] Matthias Kowal, Sofia Ananieva, and Thomas Thüm. Explaining anomalies in feature models. In *Proceedings of the 2016 ACM SIGPLAN International Conference on Generative Programming: Concepts and Experiences*, GPCE 2016, pages 132–143, New York, NY, USA, 2016. ACM.

[7] K. Lauenroth, K. Pohl, and S. Toehning. Model checking of domain artifacts in product line engineering. In *2009 IEEE/ACM International Conference on Automated Software Engineering*, pages 269–280, Nov 2009.

[8] Raul Mazo, Juan C. Munoz-Fernandez, Luisa Rincon, Camille Salinesi, and Gabriel Tamura. VariaMos: an extensible tool for engineering (dynamic) product lines. In *Proceedings of the 19th International Software Product Line Conference*, pages 374–379. ACM, 2015.

[9] Marcilio Mendonca, Moises Branco, and Donald Cowan. S.p.l.o.t.: Software product lines online tools. In *Proceedings of the 24th ACM SIGPLAN Conference Companion on Object Oriented Programming Systems Languages and Applications*, OOPSLA '09, pages 761–762, New York, NY, USA, 2009. ACM.

[10] A. Metzger, K. Pohl, P. Heymans, P. Y. Schobbens, and G. Saval. Disambiguating the documentation of variability in software product lines: A separation of concerns, formalization and automated analysis. In *15th IEEE International Requirements Engineering Conference (RE 2007)*, pages 243–253, Oct 2007.

[11] Klaus Pohl, Günter Böckle, and Frank J. van der Linden. *Software Product Line Engineering: Foundations, Principles and Techniques*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

[12] M. Pol'la, A. Buccella, M. Arias, and A. Cechich. Sevatax: service taxonomy selection validation process for spl development. In *2015 34th International Conference of the Chilean Computer Science Society (SCCC)*, pages 1–6, Nov 2015.

[13] Fabricia Roos-Frantz, José A Galindo, David Benavides, Antonio Ruiz Cortés, and J Garcıa-Galán. Automated analysis of diverse variability models with tool support. *Jornadas de Ingenierıa del Software y de Bases de Datos (JISBD 2014), Cádiz. Spain*, page 160, 2014.

[14] Fabricia Roos-Frantz, José A Galindo, David Benavides, and Antonio Ruiz-Cortés. Fama-ovm: a tool for the automated analysis of ovms. In *Proceedings of the 16th International Software Product Line Conference-Volume 2*, pages 250–254. ACM, 2012.

[15] Anjali Sree-Kumar, Elena Planas, and Robert Clariso. Analysis of feature models using alloy: A survey. In *Proceedings 7th International Workshop on Formal Methods and Analysis in Software Product Line Engineering, FMSPLE@ETAPS 2016, Eindhoven, The Netherlands, April 3, 2016.*, pages 46–60, 2016.

[16] T. von der Massen and H. H. Lichter. Deficiencies in feature models. In Tomi Mannisto and Jan Bosch, editors, *Workshop on Software Variability Management for Product Derivation - Towards Tool Support*, 2004.