Construcción y adaptación de Lenguajes de Dominio Específico por usuarios finales

Santiago Jácome G.
Universidad de las Fuerzas Armadas ESPE, Ecuador
Universidad Autónoma de Madrid, España
psjacome@espe.edu.ec

Resumen. Un Lenguaje de Domino Específico (DSL) es un lenguaje de más alto nivel que un Lenguaje de Propósito General (GPL) debido a que se logra establecer una correspondencia más estrecha entre el problema y la solución de forma sencilla y fácil de aprender. Sin embargo su construcción no es una tarea trivial ya que implica el desarrollo de varios artefactos creados por ingenieros de software expertos en modelado. La mayoría de DSLs existentes tienen una estructura rígida, debido a que el usuario no puede realizar modificaciones sobre su estructura y aspecto físico. El presente proyecto plantea desarrollar una plataforma de código abierto utilizando Eclipse Modeling Framework (EMF) que permita cargar DSLs que podrán ser adaptados por el experto en el dominio considerado éste como el usuario final de DSL, a través de técnicas de programación de usuario final. Como resultado del proyecto a más de la plataforma señalada se generará la metodología y arquitectura que la acompañan.

Palabras Clave: MDE, DSL, adaptación de DSLs, desarrollo por el usuario final.

1 Introducción

A lo largo de la pasada década la Ingeniería Dirigida por Modelos (MDE) ha surgido como un nuevo paso en el camino hacia la verdadera industrialización de la producción de software. El uso sistemático de modelos se presenta como la forma apropiada para conseguir programar con un nivel más alto de abstracción, aumentar la calidad de los productos desarrollados y sobre todo aumentar el nivel de automatización [1, 2].

El desarrollo de software con MDE en base a DSLs cubre dos etapas. La una encargada de la creación de un conjunto de artefactos (principalmente DSLs y sus mecanismos de transformación) y la otra donde personas no necesariamente con formación técnica desarrollan aplicaciones con los artefactos creados de forma simple y rápida. Esto último se vuelve posible debido a que se incrementa la facilidad de uso del entorno ya que se ofrece una vista especializada del sistema que presenta de forma precisa y coherente la información sobre la que está decidiendo [1]. Cabe señalar que el usuario de un DSL es la persona experta en el dominio que con su conocimiento,

experiencia y la herramienta adecuada puede dar la mejor solución al problema. El experto en el dominio no necesariamente debe conocer aspectos estructurales del lenguaje.

Cuando el experto en el dominio decide utilizar un DSL para resolver algún problema, tiene que utilizar el DSL que se encuentre disponible, pero ¿qué pasa cuando el usuario considera que el DSL no se ajusta plenamente a sus requerimientos?...en este caso será el experto en modelado el que tenga que realizar las modificaciones correspondientes.

Para tratar de minimizar el esfuerzo de modificación del DSL, el presente proyecto considera permitir a expertos en el dominio sin conocimientos de meta-modelado y diseño de interfaces gráficas, el poder realizar sus propias modificaciones para que se ajuste a sus necesidades particulares. Es decir, se intenta proveer al DSL cierta flexibilidad al cambio a través de técnicas de programación de usuario final, como programación por ejemplos, programación por demostración, programación visual, generación de macros, preguntas/repuestas, selección de opciones [5]. Bajo esta consideración el proyecto enlaza dos paradigmas de desarrollo de software: MDE y Desarrollo por el Usuario Final (EUD, del inglés End-User Development) el cual se define como "el conjunto de métodos, técnicas y herramientas que permiten en algún momento a los usuarios de sistemas de software, actuar como desarrolladores de software no profesionales para crear, modificar o extender un artefacto de software" [3]. Lo señalado resulta valioso en MDE porque los expertos en el dominio conocen su propio contexto y necesidades mejor que nadie y a menudo tienen conocimiento sobre la marcha de los cambios en sus respectivos dominios [4].

El resto del documento se encuentra organizado de la siguiente manera. La sección 2 señala la motivación que impulsa el desarrollo del proyecto y también se proporciona una explicación general de cómo abordar su desarrollo. En la sección 3 se describe la metodología de investigación a ser utilizada y su estrategia de evaluación. En la sección 4 se señalan algunos trabajos relacionados. Finalmente en la sección 5 se señala las conclusiones.

2 Motivación y descripción de su desarrollo

En la actualidad prácticamente todos los DSLs existentes son rígidos, es decir no permiten al experto en el dominio modificar su estructura interna y aspecto físico. Cuando se requiere realizar una modificación al DSL, ésta debe ser realizada por ingenieros de software expertos en modelado. El proyecto plantea el poder contar con una infraestructura que permita disponer de un repositorio de DSLs para múltiples propósitos como se representa en la Fig. 1. El repositorio podrá ir creciendo debido a la posibilidad de ir agregando nuevos DSLs que deberán cumplir con ciertos lineamientos establecidos para el efecto.

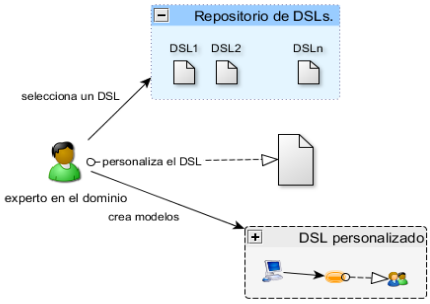

Fig. 1. Proceso de selección y personalización de DSLs.

Una vez seleccionado el DSL que más se ajuste a los requerimientos del experto en el dominio, éste podrá utilizar el DSL directamente o en su defecto podrá personalizarlo de acuerdo a sus necesidades particulares. La personalización del DSL considera algunos criterios de la tipología de variabilidad de modelos señalados en Cengarle et. al [6], tales como variabilidad de la sintaxis abstracta y/o sintaxis concreta.

En el esquema planteado cuando se requiera construir un DSL se tendrá que realizar su correspondiente análisis de dominio el cual permitirá determinar los elementos suficientes para cubrir todas sus características (obligatorias y opciones) [7]. Con la información recogida en el análisis de dominio se elaborará el modelo de características que identificará y representará las capacidades generales del dominio. A partir del modelo de características se procederá a elaborar el meta-modelo del DSL, de esta manera se podrá contar con un DSL que cubra la mayor cantidad de elementos del dominio establecido.

Posteriormente se deberá establecer una correspondencia entre los elementos del modelo de características con los elementos del meta-modelo del DSL. En esta etapa al meta-modelo se le denomina “Meta-Modelo Completo” o “MMC”. Cuando se hayan construido estos dos modelos y el mecanismo que los relacione, se procederá a desarrollar una Interfaz Gráfica de Usuario (GUI) del modelo de características.

La personalización del DSL por parte del usuario final contempla la construcción de un entorno gráfico que permitirá dos niveles de personalización que se detallan en las siguientes subsecciones.

2.1 Primer nivel de personalización del DSL

A través de la interfaz del modelo de características el experto en el dominio podrá ir seleccionando únicamente los elementos del DSL que considere, aquellos elementos no considerados serán eliminados del meta-modelo (variabilidad negativa). La selección de características de la GUI deberá ir modificando de manera transparente el meta-modelo del DSL (modificación de la sintaxis abstracta) hasta llegar a una nueva versión del meta-modelo llamada “Meta-Modelo Base” o “MMB”. Una vez que se haya definido plenamente el MMB se deberá construir con éste un DSL gráfico básico mediante algún mecanismo automatizado o semi-automatizado. Por ejemplo, si se tiene como DSL base una máquina de estados, el proceso será como el que se representa en la Fig. 2.

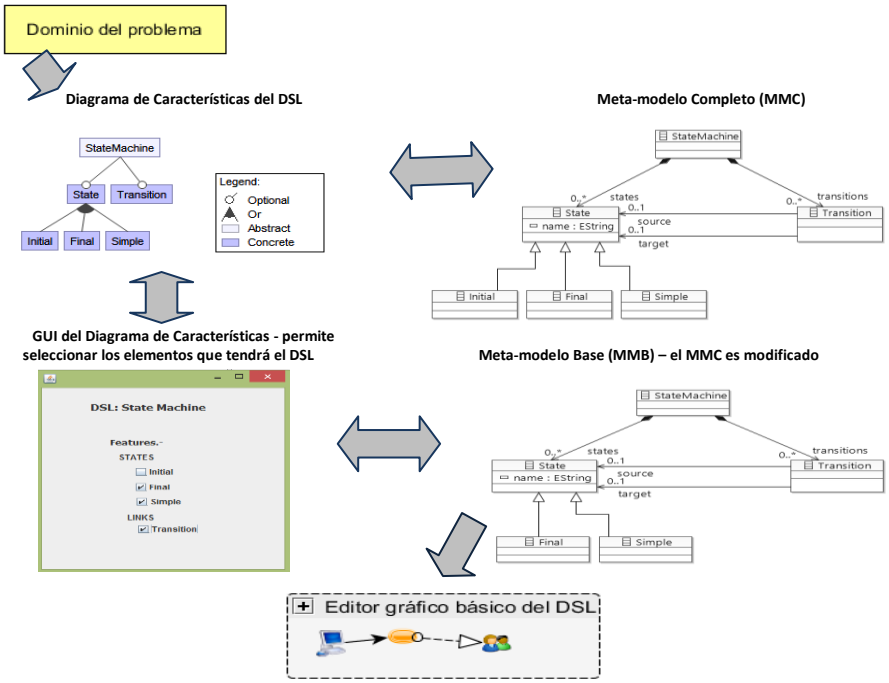

Fig. 2. Proceso de eliminación de características de la estructura del DSL.

Para desarrollar este nivel de personalización surgen varias preguntas de investigación, las cuales deberán ser respondidas y solucionadas durante la ejecución del proyecto.

Preguntas de Investigación (PI):

PI1: ¿Cuáles son los aspectos que pueden ser eliminados en el meta-modelo del DSL sin que pierda su esencia básica?
PI2: ¿Cómo establecer una correspondencia entre los elementos del modelo de características con el meta-modelo del DSL?
PI3: ¿Cuál es grado de afectación sintáctico y semántico del meta-modelo del DSL cuando se elimina una característica del modelo de características?
PI4: ¿Cuál es mecanismo más idóneo que deberá adoptar la GUI para manejar el modelo de características?
PI5: ¿Cuál es mecanismo que se puede utilizar para modificar el meta-modelo del DSL de forma transparente para el usuario?
PI6: ¿Cuál es mecanismo automático/semi-automático más adecuado para construir un DSL gráfico básico con el "meta-modelo base" del DSL?

2.2 Segundo nivel de personalización del DSL

Si el experto en el dominio considera que el DSL que va a utilizar necesita ciertos elementos no considerados, éstos podrán ser agregados por él mismo través del editor del DSL de forma fácil y amigable. Por lo que se tendrá que investigar mecanismos

para que usuarios no expertos puedan expresar nuevos requisitos sobre el lenguaje base y el entorno de modelado. En términos concretos el usuario podrá: (a) añadir nuevos elementos al DSL que pueden afectar directamente a su meta-modelo (modificación de la sintaxis abstracta) y (b) modificar la apariencia física de los elementos del DSL (modificación de la sintaxis concreta).

Este nivel de personalización del DSL también debe permitir la modificación automática y transparente del "Meta-Modelo Base" a un "Meta-Modelo Final" o "MMF". Al finalizar el proceso se contará con un meta-modelo que se ajuste a las necesidades particulares del usuario. Si se requiere mejorar el DSL gráfico se podrá utilizar el MMF como insumo para herramientas sofisticadas para el diseño de DSLs gráficos como Graphiti, GMF, Sirius, Spray o el que pueda aparecer en el futuro.

Preguntas de Investigación (PI):

PI1: ¿Cuál es el mecanismo utilizado por el usuario para añadir nuevas características al DSL de forma fácil e intuitiva?
PI2: ¿Cuál es el mecanismo que permita que el usuario conforme vaya modificando el editor gráfico del DSL se vaya modificando de forma automática y transparente su meta-modelo?
PI3: ¿Cuál es grado de afectación sintáctico y semántico del meta-modelo del DSL cuando se agreguen nuevos elementos?
PI4: ¿Cuál es mecanismo para que el experto en el dominio pueda modificar ciertas propiedades gráficas de los elementos de la paleta del editor gráfico del DSL?

3 Método de Investigación y Estrategia de Evaluación

El proyecto recae sobre la línea de investigación de tecnología, el cual será llevado a cabo por el método de "Investigación de Tecnología" propuesto por Solheim et al. [8]. "La *investigación de tecnología* es la investigación con el fin de producir nuevos y mejores artefactos" [8]. El método de investigación establece también los mecanismos para llevar a cabo la investigación y las estrategias utilizadas para su evaluación. El método considera un proceso iterativo a través de tres pasos: (a) *análisis del problema*: encontrar y definir el problema al que se necesite dar una solución, (b) *innovación*: en base a los requisitos recogidos en la etapa de análisis del problema se construye un artefacto que se supone resuelve el problema, y (c) e*valuación*: se debe validar que el artefacto realmente resuelve el problema.

El proyecto propuesto se realizará como un proceso iterativo en el cual los artefactos y los requisitos irán cambiando de acuerdo a las entradas que se tengan en el proceso conforme se vayan respondiendo a las preguntas de investigación señaladas en la sección 2. Se considera evaluar los artefactos desarrollados a través de prototipos, casos de estudios de confirmación (confirmatory case-studies) y ejemplos.

Entre los prototipos considerados a construir se encuentran: el repositorio de DSLs, entorno gráfico de personalización de DSLs, mecanismos de personalización; los mismos que serán probados en la academia y de ser posible en la industria a través de ejemplos en diversos contextos. Para la evaluación de la calidad interna de los artefactos que se vayan generando se aplicarán varias normas, estándares y estudios relacionados. En términos generales se evaluará: (a) la calidad de los meta-modelos generados [9, 10], (b) la notación visual a ser empleada en los DSLs [11], (c) la usabilidad de los plataforma de personalización de DSLs a través de métodos de

evaluación, tales como "recorrido cognitivo" (cognitive walkthrough) [12], "dimensiones cognitivas" (cognitive dimensions) [13].

4 Trabajos Relacionados

Existen varios trabajos relacionados con variabilidad de artefactos MDE, línea de productos en MDE, construcción y evolución de DSLs. Sin embargo se señalan aquellos que guardan una relación más estrecha con el presente proyecto.

Avila-García et. al [14] en el ámbito de línea de productos de software y personalización de DSL proponen un mecanismo para crear una plantilla que representa un modelo de características de una familia de modelos y que mediante transformaciones se puede especializar y configurar un producto concreto.

Rose et. al [15] desarrollan EuGENia Live, una herramienta de modelado gráfico flexible que a través de una navegador web se permite la modificación *"sobre la marcha"* del meta-modelo de uno de los DSL de un grupo de DSLs base disponibles. Posteriormente el meta-modelo en formato Emfatic puede ser exportado al entorno EMF.

Svendsen et. al [16] utilizan el Lenguaje Común de Variabilidad (CVL, del inglés Common Variability Language) para el manejo de una línea de productos de software para el dominio de señalización de trenes. CVL es un lenguaje independiente que permite definir y manejar la variabilidad de los modelos definidos con un DSL determinado para generar un modelo producto.

Jesús J. López-Fernández et. al [17] utiliza un enfoque de desarrollo de meta-modelos dirigido por ejemplos, un proceso interactivo e iterativo para la construcción de meta-modelos a través de fragmentos de ejemplo modelado por expertos de dominio bajo la tutela del experto en modelado.

Por lo tanto se puede señalar que a pesar de que se han realizado esfuerzos en el ámbito de creación y personalización de DSLs, el presente proyecto es único debido a que plantea contar con un entorno integral donde sea el mismo usuario del DSL el que realizase la personalización del DSL a través de mecanismos fáciles e intuitivos de utilizar.

5 Conclusiones

Actualmente existe muchos DSLs los cuales en su gran mayoría no permiten ser modificados en su estructura interna y apariencia física por el usuario final. Si el experto en el dominio requiere realizar cambios, será el ingeniero de software experto en modelado el que tenga que realizarlos. El presente proyecto que se encuentra en la etapa de análisis (levantamiento de requisitos) plantea el poder contar con un repositorio de DSLs "flexibles" de varios propósitos. La finalidad es poder aprovechar el conocimiento y experiencia del experto en el dominio para que sea el mismo a través de mecanismos intuitivos de programación de usuario final pueda ajustar el DSL a sus necesidades concretas.

Referencias

1. Cuevas, C., et al., *Beneficios que aporta la metodología MDE a los entornos de desarrollo de sistemas de tiempo real.* Revista Iberoamericana de Automática e Informática Industrial RIAI, 2013. 10(2): p. 216-227.
2. Mohagheghi, P. and V. Dehlen. *Where is the proof?-A review of experiences from applying MDE in industry*. in *Model Driven Architecture–Foundations and Applications*. 2008. Springer.
3. Lieberman, H., et al., *End-user development: An emerging paradigm*2006: Springer.
4. *Interacction Design Fundation.* http://www.interaction-design.org/encyclopedia/end-user_development.html
5. Costabile, M.F., et al., *End-user development: The software shaping workshop approach*, in *End user development*2006, Springer. p. 183-205.
6. Cengarle, M.V., H. Grönniger, and B. Rumpe, *Variability within modeling language definitions*, in *Model Driven Engineering Languages and Systems*2009, Springer. p. 670-684.
7. Gómez, A. and I. Ramos, *Cardinality-Based Feature Modeling and Model-Driven Engineering: Fitting them Together.* VaMoS, 2010. 37: p. 61-68.
8. Solheim, I.a.S., K., *Technology Research Explained, Technical Report A313.* 2007.
9. López-Fernández, J.J., E. Guerra, and J. de Lara. *Meta-Model validation and verification with MetaBest*. in *Proceedings of the 29th ACM/IEEE international conference on Automated software engineering*. 2014. ACM.
10. López-Fernández, J.J., E. Guerra, and J. de Lara. *Assessing the Quality of Meta-models*. in *11th Workshop on Model Driven Engineering, Verification and Validation MoDeVVa 2014*. 2014.
11. Moody, D.L., *The "physics" of notations: toward a scientific basis for constructing visual notations in software engineering.* Software Engineering, IEEE Transactions on, 2009. 35(6): p. 756-779.
12. Nielsen, J. *Usability inspection methods*. in *Conference companion on Human factors in computing systems*. 1994. ACM.
13. Green, T.R.G. and M. Petre, *Usability analysis of visual programming environments: a 'cognitive dimensions' framework.* Journal of Visual Languages & Computing, 1996. 7(2): p. 131-174.
14. Avila-García, O., A.E. García, and E. Rebull. *Using software product lines to manage model families in model-driven engineering*. in *Proceedings of the 2007 ACM symposium on Applied computing*. 2007. ACM.
15. Rose, L.M., D.S. Kolovos, and R.F. Paige. *EuGENia live: a flexible graphical modelling tool*. in *Proceedings of the 2012 Extreme Modeling Workshop*. 2012. ACM.
16. Svendsen, A., et al., *Developing a software product line for train control: A case study of cvl*, in *Software Product Lines: Going Beyond*2010, Springer. p. 106-120.
17. López-Fernández, J.J., et al., *Example-driven meta-model development.* Software & Systems Modeling, 2013: p. 1-25.