Creación Colaborativa de Lenguajes Específicos del Dominio

Javier Cánovas, Jordi Cabot

Creación Colaborativa de Lenguajes Específicos de Dominio

Javier Luis Cánovas Izquierdo, Jordi Cabot

AtlanMod, École des Mines de Nantes – INRIA – LINA, Nantes, France,
{javier.canovas,jordi.cabot}@inria.fr

Resumen El desarrollo de sofware es un proceso donde participan muchos actores, principalmente los desarrolladores y los clientes del producto. En la actualidad, procesos de desarrollo como los basados en metodologías ágiles proponen la participación de forma directa de los usuarios o clientes. La idea clave es definir procesos guiados por la comunidad donde todos los participantes (técnicos y no técnicos) colaboran para que el producto satisfaga los requisitos.
Esta aproximación es especialmente interesante en el ámbito del desarrollo de lenguajes específicos de dominio (DSL). Sin embargo, aunque estos lenguajes están destinados a una comunidad de usuarios expertos de un dominio concreto, actualmente dichos usuarios tienen poca (o nula) participación en el desarrollo. Nuestra propuesta consiste en incorporar el aspecto colaborativo en los procesos de desarrollo de DSLs, permitiendo a la comunidad de usuarios del lenguaje participar activamente en su creación y evolución. Para ello proponemos adaptar *Collaboro*, un lenguaje para representar las actividades de colaboración que surgen durante el desarrollo de DSLs, para ser utilizado a lo largo de todo el proceso.

1. Introducción

En los últimos años la mayoría de los esfuerzos para hacer más eficientes los procesos de desarrollo software se han centrado fundamentalmente en un determinado actor: los desarrolladores. El usuario final (o cliente) participa durante la elicitación de requisitos pero es ignorado hasta la etapa de pruebas, provocando normalmente que las aplicaciones no satisfagan las espectativas [1]. En respuesta a esta situación, diferentes metodologías de desarrollo, como las llamadas ágiles [2], y otros trabajos como [1,3] proponen la colaboración de los usuarios finales durante todo el proceso, permitiendo una validación continua.

El desarrollo de aplicaciones libres y de código abierto (*Free and Open Source Software*, FOSS) son el principal ejemplo de desarrollos colaborativos. Estos procesos utilizan sistemas para promover la participación como, por ejemplo, sistemas de votación (como en Apache [4]) o asignación de tareas (como en Mozilla [5]). Otro ejemplo representativo de sistemas colaborativos se puede encontrar en la web social, como los sitios web de *stackExchange*[1], donde los usuarios de la comunidad colaboran para la resolución de problemas de otros usuarios.

La participación de los usuarios finales cobra especial importancia en el desarrollo específico de dominio, en particular, en los lenguajes específicos de dominio (*Domain*

[1] http://www.stackexchange.com

*Specific Language*s, DSLs). En los DSLs, la colaboración de los usuarios se hace imprescindible ya que estos lenguajes están destinados a un dominio de aplicación del cual son precisamente expertos. Sin embargo, aunque existen un conjunto de técnicas y recomendaciones para el desarrollo de DSLs [6,7,8], la mayoría se centra en los artefactos y tareas a llevar a cabo por los desarrolladores.

Herramientas como Bugzilla[2] o Trac[3] permiten el desarrollo colaborativo de sistemas software pero no están adaptadas para tratar con DSLs. Por otro lado en trabajos como [1,3] se comenta la utilidad de la participación de los usuarios en algunas fases que utilizan modelos durante el desarrollo de software, sin embargo, no presentan la colaboración como un proceso que ayude a guiar el desarrollo. Finalmente herramientas como [9,10] solo permiten la colaboración online, lo que limita la participación y no permite la representación de las interacciones surgidas.

Nuestra propuesta consiste en hacer colaborativo el proceso de desarrollo de DSLs, conviertiéndolo en uno más democrático que tenga en cuenta las opiniones y sugerencias de la comunidad (usuarios y desarrolladores) existente alrededor del lenguaje. De forma parecida al desarrollo de FOSS y a la web social, los miembros de la comunidad pueden proponer, discutir y decidir qué cambios y soluciones son las más adecuadas. En este sentido, la colaboración se refiere a la puesta en común de propuestas y toma de decisiones conjunta. Para ello definimos un DSL, denominado Collaboro, para representar las colaboraciones de los miembros de la comunidad, es decir, las propuestas de cambio, soluciones y comentarios. Estas colaboraciones permiten llegar a acuerdos acerca de los cambios a incorporar al lenguaje así como mantener un registro de dichos cambios, permitiendo guiar el proceso de desarrollo y conocer qué motivó la creación de cada elemento, respectivamente. En [11] presentamos una primera versión del lenguaje para el desarrollo colaborativo de la sintaxis abstracta de los DSLs. En este trabajo exploramos su adaptación para el diseño de la sintaxis concreta así como otras mejoras que permitan explotar el potencial de la comunidad a lo largo de todo el proceso.

El resto del artículo está organizado de la siguiente manera. La Sección 2 contrasta un proceso de desarrollo de DSL tradicional con uno colaborativo. La Sección 3 describe Collaboro y, finalmente, la Sección 4 presenta las líneas de trabajo principales.

2. Modelo tradicional vs. colaborativo

Un DSL está compuesto de tres elementos principales [12]: sintaxis abstracta, sintaxis concreta y semántica. La sintaxis abstracta define los conceptos principales del lenguaje y sus relaciones, así como un conjunto de restricciones. La sintaxis concreta define la notación del lenguaje, que puede ser textual, gráfica o híbrida. Finalmente, la semántica del DSL se define generalmente utilizando una aproximación traslacional.

Según [6], el proceso de creación de un DSL está compuesto de cinco fases: decision, análisis, diseño, implementación y despliegue. En la fase de decisión se identifica la necesidad de crear un DSL para un dominio particular. En la fase de análisis se estudia el dominio de aplicación para, a continuación, crear el lenguaje en las fases de diseño e implementación. El lenguaje está listo para ser utilizado en la fase de despliegue.

[2] http://www.bugzilla.org
[3] http://trac.edgewall.org

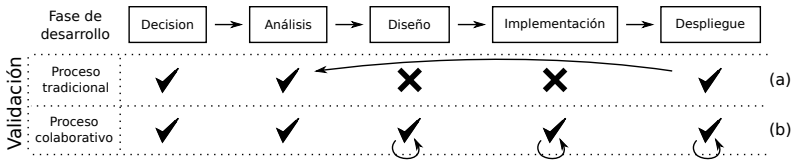

Figura 1. Validación en (a) un proceso tradicional y en (b) un proceso colaborativo.

En el modelo tradicional de desarrollo, los expertos del dominio participan en las primeras dos fases y no vuelven a colaborar hasta la última fase, donde pueden comprobar si el lenguaje cumple con sus espectativas. Así, la validación del lenguaje se retrasa hasta el final del proceso, donde pueden aparecer problemas derivados de una incorrecta interpretación de los requisitos. Esta situación normalmente implica un reinicio del proceso para resolver los problemas identificados, provocando un incremento en el coste y esfuerzo de creación del lenguaje. La Figura 1a muestra las fases del proceso e indica aquellas en las que los usuarios participan. Además, también ilustra cómo los problemas indentificados en la última fase fuerzan a un reinicio del proceso.

En un proceso en el que participa la comunidad, tanto los desarrolladores como los usuarios finales del lenguaje pueden colaborar activamente en todas las fases del desarrollo, particularmente las enfocadas al diseño e implementación. De esta forma, los usuarios ya no tienen que esperar al final del proceso para validar el lenguaje sino que cada fase es validada conforme se va realizando (ver Figura 1b). Por ejemplo, en un lenguaje para representar rutas en una empresa de transportes, una vez la comunidad de usuarios decide satisfactoriamente crear el lenguaje y se supera la fase de análisis, se procede a su diseño e implementación. En una primera versión del lenguaje, la sintaxis abstracta incluye los conceptos de *vehículo* y *ruta*, compuesta de *puntos de parada*. Por falta de espacio y dada la simplicidad del lenguaje, no mostramos el metamodelo.

En el modelo tradicional, una vez definida la sintaxis abstracta, el proceso de implementación continuaría, definiendo la sintaxis concreta del lenguaje y el resto de herramientas. Por ejemplo, la sintaxis concreta podría consistir en una representación gráfica de las rutas, puntos de parada y camiones sobre un mapa geográfico. Los usuarios finales deberían esperar hasta la fase de despliege para poder validar y detectar posibles errores en el lenguaje. Por ejemplo, los usuarios podrían considerar que una característica crucial es representar el estado de una ruta en cuestión del tráfico para poder calcular mejor qué dirección tomar. De esta forma, la ruta debería mostrarse en diferentes colores, representado el estado del tráfico. Debido a que esta característica no fue considerada en la sintaxis abstracta del lenguaje, ésta y el resto de componentes del lenguaje, como la sintaxis concreta, deberían ser actualizados para contemplar el cambio.

En un proceso colaborativo, la falta de esta característica puede ser informada por la comunidad durante el mismo proceso de definición de la sintaxis abstracta. Además, la comunidad no solo puede detectar el problema sino también proponer soluciones y decidir en conjunto cuál debería ser incorporada. Esta tarea se ve facilitada por el hecho de trabajar con un DSL de un dominio donde la comunidad es experta, permitiendo a los miembros de ésta colaborar activamente. Por ejemplo, un usuario podría informar de la necesidad de la nueva característica y, a continuación, un desarrollador podría proponer una solución para adaptar el lenguaje, la cual sería discutida, valorada y votada por

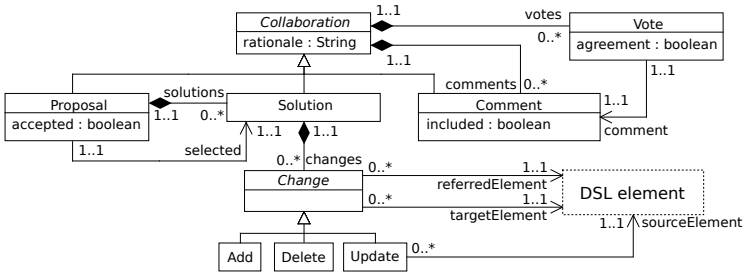

Figura 2. Parte principal del metamodelo de sintaxis abstracta de Collaboro.

la comunidad para su posterior implementación. Permitir este grado de participación requeriría poder representar las colaboraciones que surgen, lo cual permitiría además llevar un registro de todos los cambios introducidos.

3. Collaboro

Para hacer colaborativo un proceso tradicional de desarrollo de DSLs definimos Collaboro [11], un lenguaje que permite representar explícitamente las colaboraciones que surgen en la comunidad durante el proceso. Estas colaboraciones son expresadas como modelos y permiten llevar un control de los cambios aplicados durante el desarrollo del lenguaje. Collaboro, creado colaborativamente en nuestro grupo de investigación, ha sido desarrollado como un plugin de Eclipse y puede ser descargado desde su website[4].

La parte principal del metamodelo de sintaxis abstracta de Collaboro se muestra en la Figura 2 ([11] contiene una descripción más detallada). Este metamodelo permite representar información estática y dinámica. La parte estática de Collaboro representa las colaboraciones, que pueden ser: propuestas de cambio (metaclase `Proposal`), propuestas de solución (metaclase `Solution`) y comentarios (metaclase `Comment`). Cada elemento incluye una explicación en lenguaje natural (atributo `rationale`).

Las propuestas de solución describen los cambios a realizar, que pueden ser: añadir (metaclase `Add`), borrar (metaclase `Delete`) o actualizar (metaclase `Update`). Los cambios indican el elemento a modificar (referencia `referredElement`) y el elemento a cambiar (referencia `target`). En el caso de `Update`, también se indica el elemento previo al cambio (referencia `source`). Los elementos involucrados en estas referencias difieren dependiendo de la parte del DSL a cambiar, por ejemplo, pueden referir a las metaclases de la sintaxis abstracta del lenguaje que se está desarrollando.

Por otro lado, la parte dinámica registra los procesos de decisión para seleccionar aquellas propuestas de cambio y soluciones que deben incorporarse al lenguaje. Un usuario puede votar (metaclase `Vote`) a favor o en contra de una colaboración. Según los votos recibidos, una propuesta puede ser aceptada (atributo `accepted`), una solución seleccionada (atributo `selected`) o un comentario includo (atributo `included`) para alterar la propuesta/solución comentada. El proceso de decisión puede basarse en diferentes algoritmos, por ejemplo, acuerdo por mayoría simple o absoluta.

[4] http://code.google.com/a/eclipselabs.org/p/collaboro

4. Plan de trabajo. Extensión de Collaboro

Nuestro objetivo es explorar la adaptación de Collaboro para cubrir otros elementos del desarrollo de DSLs así como estudiar otras mejoras que exploten el potencial de la comunidad. A continuación describimos las líneas de trabajo identificadas.

Sintaxis concreta. Dado el número de decisiones de diseño a tratar (gráfica y/o textual, herramientas, etc), la sintaxis concreta es una de las partes más complicadas de abordar colaborativamente. Una solución para facilitar su desarrollo colaborativo sería la definición de un metamodelo básico para sintaxis textuales y gráficas, que definiría, por ejemplo, elementos como *keyword* o *figura*, respectivamente. Los cambios que impliquen modificar la sintaxis concreta se podrían representar con Collaboro (adaptando las referencias de la jerarquía `ModelChange`), permitiendo a los usuarios de la comunidad discutir a alto nivel cómo desean que se represente el lenguaje.

Definición de cambios mediante ejemplos. Actualmente los cambios a aplicar se definen según los elementos del lenguaje. Una alternativa que facilitaría esta labor sería permitir especificar los cambios a través de ejemplos del lenguaje (posiblemente inconsistentes con la última versión de éste). De esta forma, los usuarios podrían expresar qué quieren poder definir con el lenguaje. Para soportar esta extensión, nuestra aproximación debería ser capaz de derivar modelos Colloboro a partir de los ejemplos.

Visualización de los cambios. Para facilitar la comprensión de las propuestas de solución así como ayudar a la decisión de su aceptación, sería interesante disponer de un generador de ejemplos de modelos. De esta forma, cualquier usuario de la comunidad podría visualizar cómo afectarían los cambios propuestos en la solución.

Políticas de decisión. La implementación de diferentes políticas de decisión permitiría adaptar la colaboración a la comunidad, por ejemplo, ponderando los votos de acuerdo a qué usuario está votando. Por otro lado, creemos que la formalización de las discusiones asociadas a los cambios, representadas actualmente en lenguaje natural, podrían enriquecer los procesos de decisión. Finalmente, el análisis de las decisiones adoptadas serviría de fuente de información para identificar patrones de calidad en lenguajes.

Extensión de soporte de la sintaxis abstracta. Aunque el desarrollo de la sintaxis abstracta ya ha sido abordado en la primera implementación de Collaboro, todavía no se ha abordado la definición de sus restricciones. Esta definición se realiza normalmente por medio de un lenguaje declarativo como OCL, lo cual normalmente requiere de conocimientos avanzados. Dado que las expresiones OCL pueden representarse como modelos, nuestra primera propuesta consiste en adaptar Collaboro para permitir que los cambios descritos en las soluciones puedan tratar con dichos modelos (adaptando también la jerarquía `ModelChange`). De esta forma, los expertos en OCL de la comunidad integrarían las restricciones expresadas por usuarios. Más adelante nos gustaría estudiar la posibilidad definir estas restricciones con algún lenguaje que facilite la colaboración de todos los miembros de la comunidad.

Automatización de la generación del DSL. Con el soporte a la sintaxis concreta, los modelos Collaboro podrían especificar las relaciones entre las sintaxis abstracta y concreta, permitiendo generar la información necesaria por herramientas de definición de DSLs para la implementación del lenguaje. Estas herramientas normalmente requieren una definición de elementos de notación y una definición de correspondencias entre

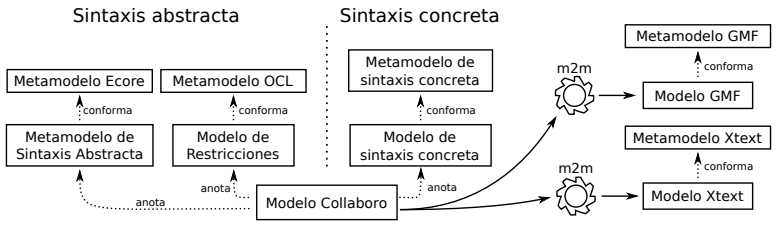

Figura 3. Aplicación de Collaboro para las sintaxis y generación de un DSL.

dichos elementos y los conceptos de la sintaxis abstracta. Dado que estas dos definiciones se representan generalmente como modelos (este es el caso de herramientas como XText[5] y GMF[6] para DSLs textuales y gráficos, respectivamente), los modelos Collaboro podrían ser transformados para obtener los modelos de la herramienta destino. La Figura 3 muestra el uso de Colloboro con los modelos involucrados en la definición colaborativa de las sintaxis concreta y abstracta así como las transformaciones de modelo requeridas, por ejemplo, para GMF y/o Xtext, para generar el DSL.

Sintaxis para Collaboro. Actualmente la manipulación de modelos Collaboro se realiza en el entorno Eclipse pero nos gustaría explorar otras alternativas para facilitar el acceso a los usuarios, por ejemplo, con una sintaxis textual o integrándolo en la web.

Referencias

1. Hildenbrand, T., Rothlauf, F., Geisser, M., Heinzl, A., Kude, T.: Approaches to collaborative software development. In: CISIS Conference, IEEE (2008) 523–528
2. Agile Manifesto. http://agilemanifesto.org
3. Whitehead, J.: Collaboration in software engineering: A roadmap. In: FOSE Conference, IEEE (2007) 214–225
4. Fielding, R.T.: Shared leadership in the apache project. Commun. ACM **42** (1999) 42–43
5. Mockus, A., Fielding, R.T., Herbsleb, J.D.: Two case studies of open source software development: Apache and mozilla. ACM Trans. Softw. Eng. Methodol. **11** (2002) 309–346
6. Mernik, M., Heering, J., Sloane, A.M.: When and how to develop domain-specific languages. ACM Comput. Surv. **37** (2005) 316–344
7. Kelly, S., Pohjonen, R.: Worst practices for domain-specific modeling. IEEE Software **26**(4) (2009) 22–29
8. Völter, M.: MD*/DSL best practices. http://voelter.de/data/pub/dslbestpractices-2011update.pdf
9. Brosch, P., Seidl, M., Wieland, K., Wimmer, M., Langer, P.: We can work it out: Collaborative conflict resolution in model versioning. In: ECSCW, Springer (2009) 207–214
10. Gallardo, J., Bravo, C., Redondo, M.A.: A model-driven development method for collaborative modeling tools. Network and Computer Applications **35** (2012) 1086–1105
11. Cánovas Izquierdo, J.L., Cabot, J.: Community-driven language development. In: MiSE Workshop, IEEE (2012) 29–35
12. Kleppe, A.: Software Language Engineering: Creating Domain-Specific Languages Using Metamodels. Addison Wesley (2008)

[5] http://www.eclipse.org/Xtext
[6] http://www.eclipse.org/gmf