III CADI
IX CAEDI
2016
confedi
UNIVERSIDAD TECNOLÓGICA NACIONAL FACULTAD REGIONAL RESISTENCIA
UNIVERSIDAD NACIONAL DEL NORDESTE FACULTAD DE INGENIERÍA

DESARROLLO DE SOFTWARE DIRIGIDO POR MODELOS EN ENTORNOS ÁGILES

Verónica Andrea Bollati, CONICET - UTN, vbollati@gmail.com
Silvio Gonnet, INGAR (CONICET – UTN), sgonnet@santafe-conicet.gov.ar
Liliana Cuenca Pletsch , UTN, cplr@frre.utn.edu.ar
Horacio Leone, INGAR (CONICET – UTN), hleone@santafe-conicet.gov.ar

Resumen— Las prácticas de desarrollo ágil se han consolidado como un estándar de la industria en los últimos años. Su piedra fundamental fue plantada por profesionales, que en 2001, realizaron un manifiesto donde propusieron cuatro valores y doce principios. Usando como base dicho manifiesto surgieron distintas iniciativas como: eXtreme Programming, Crystal Family, o Scrum y algunas más actuales como Integración y Entrega Continua, DevOps, Managment 3.0 o Lean.

Actualmente existen reportes de la industria que demuestran una creciente adopción de estas prácticas en equipos de diverso tamaño y características. Sin embargo, muchas de estas técnicas no tienen definido completamente sus procesos y mucho menos existen herramientas que brinden soporte a dichos procesos.

Proponemos aplicar las técnicas de la Ingeniería Dirigida por Modelos (MDE) al modelado de dichos procesos y a la posterior construcción de herramientas que los soporten. Para ello se plantea, aplicando MDE, definir un marco para soportar el modelado de los procesos ágiles, definiendo nuevos DSL, editores, transformaciones y generando modelos de trazas. Para la validación, además de las herramientas de desarrollo, se llevarán a cabo casos de estudio reales en colaboración con empresas del medio e internacionales.

En este trabajo se presenta, el problema a resolver, la metodología a seguir y los principales resultados de la investigación hasta este momento.

Palabras clave— *Prácticas Ágiles, Ingeniería Dirigida por Modelos, SCRUM, Integración Continua, Entrega Continua, Lean, Managment 3.0.*

1. Introducción

La Ingeniería Dirigida por Modelos (*Model Driven Engineering*, MDE) es un paradigma de desarrollo software cuyas principales características son: potenciar el papel de los modelos y las actividades de modelado en las diferentes etapas del proceso de desarrollo y aumentar el

nivel de automatización, mediante la construcción de herramientas de soporte, en cualquier actividad relacionada con el desarrollo [1],[2].

En realidad, el nacimiento de este paradigma a finales del 2000, constituyó un nuevo paso en la tendencia histórica a elevar el nivel de abstracción al que se concibe y desarrolla el software: los lenguajes de ensamblador dieron lugar a la programación estructurada que dio paso a la orientación a objetos, etc.

Esta idea se plasma en el Desarrollo de Software Dirigido por Modelos (*Model-Driven Software Development*, MDSD) [3], una de las formas más adoptadas de poner en práctica los principios de la MDE, con el objetivo final de producir software. Además, bajo la premisa de que todo es un modelo [4], MDE ha comenzado a tener una influencia directa sobre otras disciplinas de Ingeniería del Software, como la Orientación a Servicios [5] o Web Engineering [6, la domótica [7] o la ingeniería de rendimiento [8].

En los últimos años, han ido ganando protagonismos los métodos de desarrollo ágil, o prácticas ágiles, que surgieron inicialmente como reacción de la comunidad a las metodologías pesadas que tradicionalmente han dominado el mundo de la Ingeniería de Software. La piedra fundamental de esta corriente fue plantada por un grupo de profesionales, que en 2001, decidieron realizar un manifiesto (llamado Manifiesto Ágil) en cual propusieron cuatro valores y doce principios [9]. Estos principios y valores, a primera vista radicales, fueron recibidos con aceptación por parte de la comunidad de desarrolladores y con cierto escepticismo por parte de la comunidad académica. Usando como base dicho manifiesto se propusieron distintas iniciativas. Dentro de las más antiguas y difundidas se pueden citar: *eXtreme Programming* [10], [11] propuesta originalmente por Kent Beck, la familia de metodologías *Crystal Family* [12], [13] propuesta por Alistair Cockburn, o *Scrum* [14] propuesto por Ken Schwaber y Jeff Sutherland y otras que han cobrado mucha relevancia en la actualidad como: Integración Continua [15], que propone organizar el equipo de desarrollo en torno a herramientas que permitan integrar y generar versiones de software de manera continua; Entrega Continua [16], que se basa en la anterior y que permite al equipo comercial disponer de versiones listas para ser desplegadas en producción en cualquier momento; *DevOps* [17], que aboga por disminuir la brecha existente entre los equipos de desarrollo y de operaciones en las organizaciones; *Managment 3.0* [18], [19], [20], que propone una serie de prácticas para motivar y liderar equipos de desarrollo de software o *Lean* [21], que propone la eliminación del proceso de creación de software de todos aquellos procesos que no generen valor para el cliente.

Más allá de las discusiones suscitadas originalmente por este cambio radical en el modo de gestionar proyectos de desarrollo de software, existen reportes de la industria que demuestran una creciente adopción de estas prácticas y metodologías en equipos de diverso tamaño y características [22]. Es por ello que a día de hoy es necesario conducir esfuerzos de investigación que permitan realizar una mejor utilización de ellas.

Sin embargo, actualmente muchas de éstas técnicas no tienen definido completamente los procesos y mucho menos existen herramientas que brinden soporte a los procesos propuestos.

Por ello como parte del trabajo que aquí se presenta, se plantea aplicar las técnicas de MDE al modelado de dichos procesos y a la posterior construcción de herramientas que los soportan.

El resto del artículo se estructura como sigue: en la sección 2 se presenta el método de investigación que se seguirá para llevar a cabo la propuesta. La sección 3 detalla la propuesta a realizar, en la sección 4 se enumeran los resultados esperados. Por último, en la sección 5, se describen las principales conclusiones.

2. Método de Investigación

El método a utilizar se basa en una adaptación del método denominado *Investigación en Acción*. Éste es un método cualitativo utilizado para validar los trabajos de investigación mediante su aplicación a proyectos reales. En la Conferencia sobre Procesamiento de Información de 1998 se celebró la aceptación de métodos cualitativos como métodos de investigación apropiados para el campo de los sistemas de información [23], y han merecido la atención de la revista IEEE *Transactions on Software Engineering* en su número especial sobre Ingeniería del Software Empírica [24].

De un modo muy resumido, la Investigación en Acción consiste en un proceso que se basa en la aplicación, en un proyecto real, de los resultados de investigación paralelamente al propio proceso de investigación. De este modo, el proyecto real permite detectar problemas no resueltos y los investigadores pueden proponer resultados cuya validez puede comprobarse mediante su aplicación a este mismo proyecto. El proceso definido por Investigación en Acción no es un proceso lineal, sino que va avanzando mediante la compleción de ciclos. Al comenzar cada ciclo se ponen en marcha nuevas ideas, que son puestas en práctica y comprobadas hasta el inicio del siguiente ciclo [25], tal como se muestra en la Figura 1. Este proceso cíclico, en el que iremos probando y refinando cada uno de los resultados obtenidos será nuestro modo de validación.

Según Wadsworth [25], en *Investigación en Acción* existen cuatro tipos de participantes, que pueden coincidir en algunas ocasiones. Estos participantes, en el proyecto que se presentan serán:

- El **investigador**, que es aquel que impulsa como sujeto el proceso investigador. En este caso, los investigadores son los miembros del grupo de investigación en el contexto del proyecto PIP-112-201101-00906.
- El **objeto investigado**, que es el problema a resolver. En nuestro caso: la aplicación de técnicas y herramientas de Ingeniería Dirigida por Modelos a procesos de las técnicas de desarrollo ágiles de Ingeniería de Software.
- **Aquél para quien se investiga**, en el sentido del que tiene un **problema que necesita ser resuelto y que participa en el proceso investigador**. La investigación planteada trata de resolver un problema que afecta directamente a la comunidad de desarrolladores de software y concretamente de sistemas de información. En este trabajo, se contará, inicialmente, con la Empresa Essentit, que proporcionará casos de estudios reales sobre los que se podrán probar los avances realizados. Durante la ejecución de este trabajo se tratará de ampliar el universo de empresas y para ello se

ha iniciado gestiones con el Polo IT Chaco con el fin de realizar transferencia tecnológica a sus socios.

- **Aquél para quien se investiga**, en el sentido que puede **beneficiarse del resultado de la investigación,** aunque no participe directamente en el proceso. Serían, tanto los usuarios informáticos que pudieran beneficiarse de la utilización de las soluciones propuestas, como los usuarios de las aplicaciones desarrolladas como casos de estudio.

Pero además de este método, que permitirá abordar las principales problemáticas del proyecto, así como la validación de los principales resultados del mismo, a lo largo de su realización necesitaremos aplicar otro tipo de métodos específicos para resolver y validar problemas concretos. Así, por ejemplo, la validación de los procesos y herramientas relacionadas con el *Management 3.0* muy probablemente deberá apoyarse en otros métodos cualitativos que permitan valorar también factores humanos (por ejemplo, entrevistas).

Este es sólo un ejemplo de problemas que requerirán métodos de validación concretos, pero nos encontraremos con otro tipo de cuestiones como son la validación de los modelos, la validación de las transformaciones entre modelos, etc. Cada uno de estos problemas requerirá de un tipo de validación y de métodos específicos (validación formal o empírica).

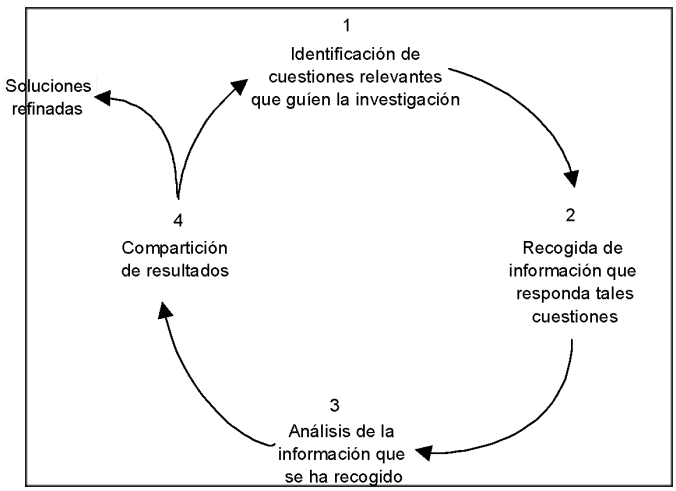

Figura. 1. Proceso cíclico de Investigación en Acción

Por otro lado, es importante mencionar el método que se utilizará para realizar el análisis del estado del arte de cada una de las técnicas y prácticas ágiles a investigar, para esto realizará una revisión sistemática, siguiendo el método propuesto por Kitchenham [26]. Este método sirve para identificar, evaluar e interpretar toda la información relativa a un tema de investigación en particular, de un modo sistemático y replicable. Surge de la investigación en el campo de la medicina, por lo que, según sus autores [26], se ha convertido en una metodología confiable, rigurosa y auditable.

La aplicación de las revisiones sistemáticas en el ámbito de la Ingeniería de Software permite dar un valor científico a la revisión de la literatura que se hace, definir una estrategia de búsqueda de la literatura a evaluar y obtener finalmente hipótesis a favor o en contra de dicha literatura. Para el desarrollo de este trabajo de investigación se ha tomado como

referencia la adaptación del método de revisiones sistemáticas para Ingeniería de Software presentado por Biolchini et. al. en [27]. En este trabajo se propone una nueva aproximación, en la cual el proceso de las revisiones sistemáticas está compuesto por cuatro grandes fases: planificación, ejecución, análisis de los resultados y resguardo de los resultados obtenidos (Figura 2).

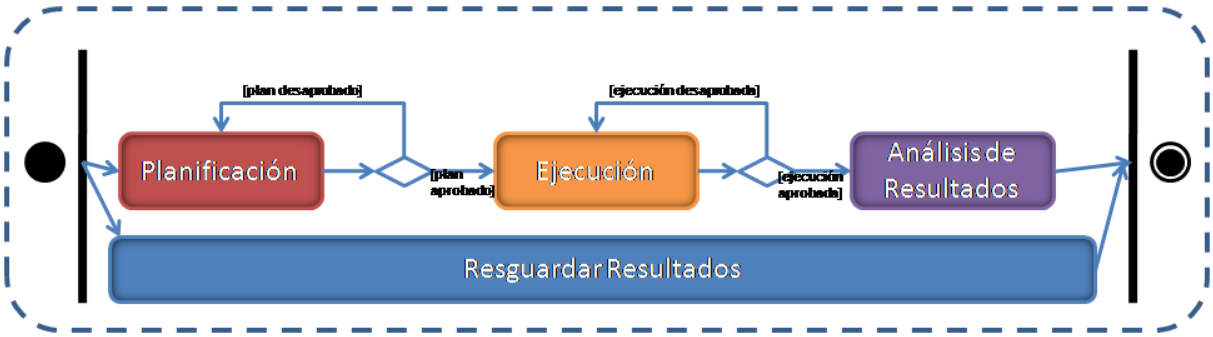

Figura 2. Proceso de Método de Revisiones Sistemáticas

En la fase de **planificación** se debe establecer claramente el objetivo de la investigación y definir el protocolo de revisión a utilizar. Es decir, definir un protocolo para cada objeto a ser investigado, estableciendo el método que será utilizado a lo largo de la realización de la revisión. Además se deben identificar los criterios de inclusión y exclusión que se seguirán para determinar las fuentes de investigación y los estudios (o documentos) a seleccionar.

En la fase de **ejecución** se lleva a cabo lo planificado en la etapa anterior, por lo que en primer lugar, se debe determinar el conjunto de estudios a evaluar. Estos estudios se seleccionan a través de la evaluación, para cada uno de ellos, de los criterios de inclusión y exclusión determinados anteriormente.

En la fase de **análisis de resultados** se debe sintetizar y evaluar la información extraída de cada estudio.

Por último, se debe mencionar que la fase de **resguardo de resultados** se realiza durante todo el proceso de la revisión sistemática, ya que a medida que se ejecutan cada una de las fases, el resultado de las mismas debe ser almacenado.

Para la realización de esta actividad se planea realizar una revisión sistemática por cada uno de las técnicas ágiles que se analizarán. De esta manera acotaremos el objeto de estudio de cada una de las revisiones, permitiendo hacer un análisis exhaustivo del material encontrado.

3. Un enfoque MDE para el modelado de procesos ágiles

Durante los últimos años, el impacto de MDE ha logrado la atención de la comunidad de Ingeniería del Software, lo que ocasionó el surgimiento de numerosas propuestas metodológicas basadas en MDE. Como lo hemos mencionado previamente, una de las que más adeptos tiene es el Desarrollo de Software Dirigido por Modelos (DSDM) [28], que consiste básicamente en la aplicación de los principios de MDE a problemas de Ingeniería del Software El punto de partida es una especificación del sistema lo más completa y precisa

posible recogida en uno o varios modelos de alto nivel. Estos modelos son sucesivamente refinados en modelos de más bajo nivel que mapean los conceptos y abstracciones recogidos en la especificación a elementos y componentes de la plataforma tecnológica de destino. Finalmente, el nivel de detalle de estos modelos de bajo nivel permite generar de forma automática el código fuente del sistema (y no sólo un esqueleto como sucedía tradicionalmente). Estos refinamientos sucesivos se implementan por medio de transformaciones de modelos.

Así, tal y como muestra la Figura 3, una vez elaborada la especificación de partida, el resto del proceso de desarrollo puede verse como un conjunto de pasos en los que uno o más modelos son tomados como entrada para generar uno o más modelos de salida.

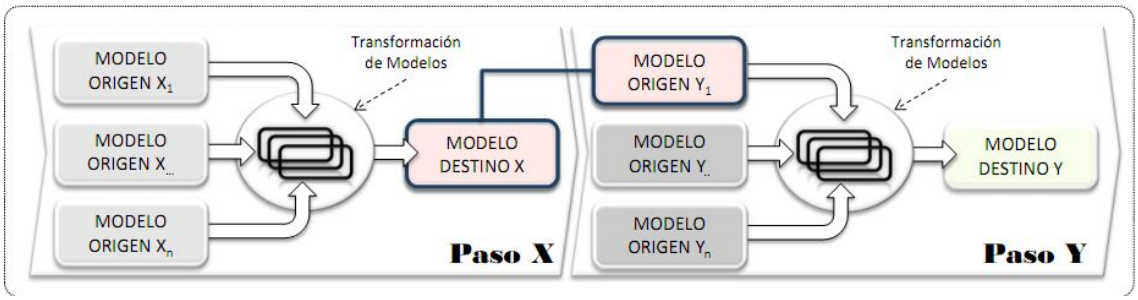

Figura 3. Vista simplificada de un proceso de desarrollo dirigido por modelos

En los pasos subsiguientes los modelos generados serían utilizados como entradas y así sucesivamente hasta que se obtiene el código que implementa el sistema. Nótese que en realidad el propio código no es más que otro modelo, pero éste con un menor nivel de abstracción, sólo el suficiente para ocultar los detalles del sistema operativo subyacente. Por todo ello, es evidente que las transformaciones de modelos son la pieza clave en cualquier propuesta de desarrollo que siga los principios de MDE.

La única forma de hacer realidad los teóricos beneficios de MDE pasa por elevar el nivel de automatización [29], lo que ha dado lugar a la aparición de un elevado número de herramientas que brindan soporte a algunas de las tareas relacionadas con MDE. Por ejemplo, como MDE se basa en el uso de modelos, se pueden encontrar herramientas para definir y utilizar nuevos lenguajes de modelado, como ATOM^3 [30], las *DSL Tools* de Microsoft [31] (reformuladas como *Visualization and Modeling Feature Pack for Visual Studio*) o el *Eclipse Modelling Framework* (EMF) [32]; para poder *conectar* dichos modelos existen varios lenguajes de transformación, como ATL (*Atlas Transformation Language*) [33], ETL (*Epsilon transformation Language*) [34], o los diferentes intentos por implementar el estándar QVT (*Query View Transformation*) [35]; para traducir estos modelos a código existen herramientas como Acceleo o XPand, incluso comienzan a aparecer herramientas para dar soporte a tareas más avanzadas, como los lenguajes TCS [36] y Gra2MoL [37] para la extracción de modelos.

Sin embargo, en muchos casos, aplicar los principios de MDE implica el desarrollo de nuevos lenguajes de modelado, normalmente como Lenguajes Específicos de Dominio,

(*Domain Specific Languages*, DSL) [38] y la generación de herramientas para trabajar con esos DSLs: editores, transformaciones, generadores de código, etc.

Siguiendo esta línea, en este trabajo se pretende aplicar los principios de MDE al modelado de los procesos propuestos en las técnicas de desarrollo de software ágiles como: Integración Continua, Entrega Continua, Managment 3.0, DevOps, Lean y, a partir de dichos modelos, construir herramientas que soporten los procesos, para esto será necesario definir nuevos DSL, editores, transformaciones y generar modelos de trazas.

Para poder aplicar los principios de MDE al modelados de los procesos propuestos en las diferentes técnicas de desarrollo de software ágiles se propone una serie de pasos a realizar (Figura 4):

1. En primer lugar se realizará un **análisis** de cada una de las técnicas, con el objetivo de identificar el conjunto de procesos que la componen.
2. Luego, a partir de esta identificación se aplicarán las técnicas del MDE con el objetivo de modelizar cada uno de estos procesos. Para esto, se **especificará un DSL** para el modelado de cada uno de los procesos, centrado en la descripción de su interfaz e independiente de la tecnología concreta con la que se implementen.
3. A continuación, se **construirá un entorno de trabajo** que soporte dicho DSL y que integrará editores de modelos y validadores. Se implementarán una serie de transformaciones, que permitirán la generación de modelos que podrán ser luego procesados también usando técnicas de MDE.
4. Por último, se realizará la **validación de los modelos y herramientas** desarrolladas por medio del desarrollo de un caso de estudio real en colaboración con las empresas que se utilicen para relevar los mismos. La validación, al igual que en las actividades anteriores, se realizará mediante un proceso iterativo con retroalimentación continua entre los resultados de investigación y el desarrollo de la herramienta de soporte y del caso de estudio.

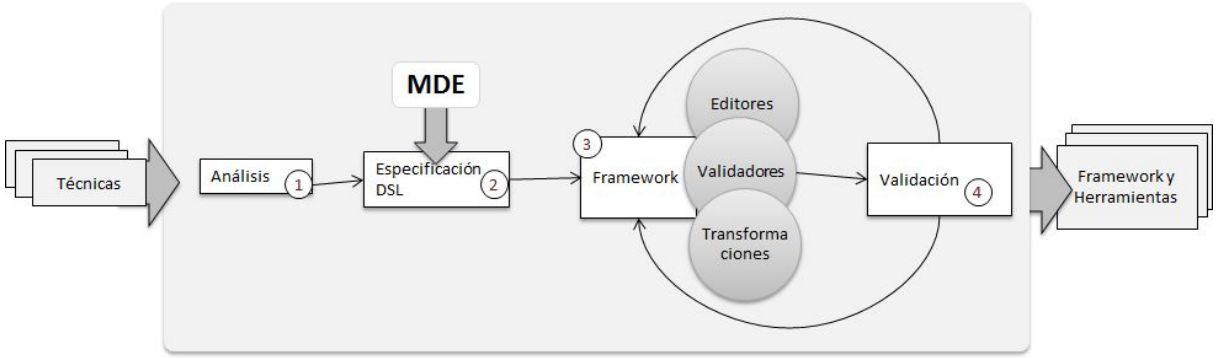

Figura 4. Vista simplificada del proceso de construcción de las herramientas

Uno de los principales problemas detectados a priori, es que la mayoría de las empresas de la región que comienzan a incursionar en el desarrollo de software ágil tienen problemas a la hora de tratar de implantar las diferentes prácticas propuestas. Desde este punto de vista, el trabajo que se presenta, busca facilitar, mediante la definición de modelos y herramientas, la adopción de las diferentes prácticas en el día a día de las empresas de desarrollo de la región haciendo más eficiente la gestión de sus proyectos de software. Para esto, de manera paralela al análisis de cada una de las técnicas y el posterior modelado de los procesos se llevará a

cabo un estudio del estado de la práctica en las empresas de la región, para poder determinar posibles puntos de mejora de las prácticas y técnicas que utilizan.

4. Resultados esperados

El principal resultado de este trabajo es un entorno para el desarrollo de software siguiendo las prácticas ágiles.

Este resultado aportará a la Ingeniería del Software como campo de conocimiento orientado al desarrollo de metodologías y herramientas tendientes a la utilización de procesos ingenieriles que atiendan la particularidad del software como producto intelectual.

En particular busca aportar a un área más específica como es el desarrollo de software mediante técnicas y prácticas ágiles, mediante la definición de modelos y herramientas que faciliten la adopción de las diferentes prácticas en el día a día de las empresas de desarrollo de la región haciendo más eficiente la gestión de proyectos de software y mejorando la productividad de sus equipos de desarrollo, lo que permitirá aumentar la competitividad de dichas empresas.

Para promover la adopción de los resultados por parte de las empresas de desarrollo de la región se propone realizar capacitación a los directivos y personal de las mismas, mostrar demos de los productos, realizar trabajo in situ en algunas empresas para modelar los procesos y desarrollar las herramientas en forma conjunta.

5. Conclusiones y recomendaciones

En este trabajo se presenta el plan de trabajo para la construcción de un entorno, aplicando los principios de MDE, para el desarrollo de software siguiendo las prácticas ágiles.

Como hemos dicho previamente, MDE es una tendencia reciente en la Ingeniería de Software cuyos principios principales son para mejorar la función de los modelos y aumentar el nivel de automatización en cualquier etapa del proceso de desarrollo [39]. La idea principal de MDE es la de considerar a los modelos como entidades de primera clase, al igual que las clases son el bloque de construcción básico en la programación orientada a objetos, o los componentes de software son la unidad básica en Ingeniería de Software basado en componentes. De hecho, MDE es un paso natural en la tendencia histórica de la Ingeniería de Software para elevar el nivel de abstracción en el que el software está diseñado y desarrollado.

Por otro lado, en los últimos años, han ido ganando protagonismos los métodos de desarrollo ágil y han surgido distintas iniciativas que ponen en práctica sus principios. Sin embargo, actualmente muchas de éstas técnicas no tienen definido completamente los procesos y mucho menos existen herramientas que brinden soporte a los procesos propuestos. Por ello, en este trabajo se plantea aplicar las técnicas de MDE al modelado de dichos procesos y a la posterior construcción de herramientas que los soportan, para esto será necesario definir nuevos DSL, editores, transformaciones y generar modelos de trazas.

Agradecimientos

Este trabajo ha sido financiado en forma conjunta por CONICET y la Universidad Tecnológica Nacional. Se agradece el apoyo brindado por estas instituciones.

6. Referencias

[1]. Bézivin, J. (2004). In search of a Basic Principle for Model Driven Engineering. Novatica/Upgrade, V(2), 21-24.
[2]. Schmidt, D.C. (2006) Model-Driven Engineering. IEEE Computer 39, 2 (2006) 25–31.
[3]. Stahl, T., Volter, M., & Czarnecki, K. (2006). Model-Driven Software Development: Technology, Engineering, Management. John Wiley & Sons.
[4]. Bezivin, J. (2004). In search of a Basic Principle for Model Driven Engineering. Novatica/Upgrade, V(2), 21-24.
[5]. Bell, M. Service-Oriented Modeling (SOA): Service Analysis, Design, and Architecture. Wiley, Hoboken, NJ, USA (2008).
[6]. Koch, N., Meli, S., Moreno, N., Pelechano, V., Snchez, F. and Vara, J.M. Model-Driven Web Engineering, UPGRADE, 9(2). Pp.40-45.
[7]. Jimenez, M., Rosique, F., Sanchez, P., Alvarez, B., and Iborra, A. (2009). Habitation: A Domain-Specic Language for Home Automation. IEEE Software, 26(4). 30-38.
[8]. Boonma, P. and Suzuki, J. 2010. Moppet: A Model-Driven Performance Engineering Framework for Wireless Sensor Networks. Computer Journal, 53(10), 1674-1690.
[9]. Manifesto for Agile Software Development, Accesible en: www.agilemanifesto.org, Utah (2001)
[10]. Extreme Programming http://www.extremeprogramming.org/
[11]. Beck, K. (1999). Extreme programming explained: Embrace change. USA. Addison-Wesley Professional.
[12]. Alistair Cockburn; "Crystal Clear, A Human-Powered Methodology for Small Teams"; October 2004, Addison-Wesley Professional, ISBN 0-201-69947-8
[13]. Cockburn, A.: Agile Software Development: The Cooperative Game, Addison-Wesley Professional (2006)
[14]. Sutherland, J., Schwaber, K., "SCRUM Development Process; Business Object Design and Implementation". 10th Annual Conference on Object-Oriented Programming Systems, Languages, and Applications Addendum to the Proceedings. ACM/SIGPLAN October, 1995
[15]. Duvall, P. M. (2006). Continuous integration: Improving software quality and reducing risk Addison-Wesley.
[16]. Humble, J., & Farley, D. (2010). Continuous delivery: Reliable software releases through build, test, and deployment automation Addison-Wesley.
[17]. Debois, Patrick (2009). "DevOps Days Ghent". DevopsDays. Retrieved 31 March 2011.
[18]. Appelo, J. Management 3.0, Leanding Agile Developers, Developing Agile Leaders. ISBN: 13:978-0-321-71247-9. (2011).
[19]. DeMarco, T., Lister, T. Peopleware. Productive Projects and Teams. 3º Ed. ISBN: 13:978-0-321-93411-6 (2013).
[20]. Fitzpatrick, B., Collins-Sussman, B. Team Geek. 1º Ed. ISBN: 13-978-144-930244-3. (2012).

[21]. Poppendieck, M., Poppendieck, T. Lean Software Development: An Agile Toolkit Addison-Wesley Professional (2003).
[22]. Agile Project Success Rates Survey Results, 2010; http://www.ambysoft.com/surveys/agileSuccess2010.html
[23]. Avison, D., Lan, F., Myers, M. and Nielsen, A. (1999). Action Research. Communications of the ACM.
[24]. Seaman, C.B. (1999). Qualitative Methods in Empirical Studies of Software Engineering. IEEE Transactions on Software Engineering, 25, 4, pp. 557-572.
[25]. Wadsworth, Y. (1998). What is Participatory Action Research? Action Research International. Accedido el 15/1/2001 en: http://www.scu.edu.au/schools/sawd/ari/ari-wadsworth.html.
[26]. Kitchenham, B. *Procedures for Performing Systematics Reviews*. Keele University Technical Report TR/SE-0401. ISSN:1353-7776. NICTA Technical Report 0400011T.1. July, 2004
[27]. Biolchini, J., Gomes Mian, P., Cruz Natali, A.C., Horta Travasso, G. *Systematic Review in Software Engineering.* Technical Report ES 679/05. May, 2005.
[28]. Stahl, M. Volter, and K. Czarnecki, € Model-Driven Software Development: Technology, Engineering, Management. Hoboken, NJ, USA: Wiley, 2006.
[29]. Atkinson, C., & Kuhne, T. (2003). Model-driven development: a metamodelling foundation. IEEE Software, 20(5)
[30]. De Lara, J., Vangheluwe, H. & Alfonseca, M. (2004). Meta-Modelling and Graph Grammars for Multi-Paradigm Modelling in AToM3, Journal on Software and Systems Modelling, Vol 3(3).
[31]. Cook, S., Jones, G., Kent, S., & Cameron Wills, A. (2007). Domain-Specific Development with Visual Studio DSL Tools. Addison-Wesley Professional.
[32]. Budinsky, F., Merks, E., & Steinberg, D. (2008). Eclipse Modeling Framework 2.0 (2nd Edition): Addison-Wesley Professional.
[33]. Jouault, F., Allilaire, F., Bézivin, J., & Kurtev, I. (2008). ATL: A model transformation tool. Science of Computer Programming, 72(1-2), 31-39.
[34]. Kolovos, D., Paige, R., & Polack, F. (2008). The Epsilon Transformation Language. Theory and Practice of Model Transformations, 5063 LNCS, 46-60.
[35]. OMG. MOF 2.0 Query/View/Transformation (QVT), V1.0. OMG Document - formal/08-04-03.
[36]. Jouault, F., Bezivin, J. and Kurtev, I. (2006). TCS: a DSL for the Specification of Textual Concrete Syntaxes in Model Engineering. In: GPCE'06: Proceedings of the fifth international conference on Generative programming and Component Engineering, Portland, Oregon, USA, pages 249—254.
[37]. Cánovas, J-L. and García Molina, J. 2009. A Domain Specific Language for Extracting Models in Software Modernization. In Proceedings of the 5th European Conference on Model Driven Architecture - Foundations and Applications. Springer-Verlag, Berlin, Heidelberg, 82-97.
[38]. Mernik, M., Heering, J., & Sloane, A. M. (2005). When and how to develop domain-specific languages. ACM Computer Surveys, 37(4), 316-344.
[39]. D.C. Schmidt, Model-driven engineering. IEEE Computer 39 (2), (2006) 25-31.

Anexo: Listado de términos y siglas

ATL	Atlas Transformation Language
EMF	Eclipse Modelling Framework
ETL	Epsilon transformation Language
DSDM	Desarrollo de Software Dirigido por Modelos
DSL	Domain Specific Languages - Lenguajes Específicos de Dominio
MDE	Model Driven Engineering - Ingeniería Dirigida por Modelos
MDSD	Model-Driven Software Development
QVT	Query View Transformation