Director
Teoría de la Computación

Ulises C. Ramirez [ulir19@gmail.com, ulisescolrez@gmail.com]
Héctor J. Chripczuk [hectorejch@gmail.com]

01 de Noviembre, 2018

Versionado

Para el corriente documento se está llevando un versionado a fin de mantener un respaldo del trabajo y además proveer a la cátedra o a cualquier interesado la posibilidad de leer el material en la última versión disponible.

Repositorio: *https://github.com/ulisescolina/UC-TC*

–Equipo Director

Índice general

Capítulo 1

Estado de la Cuestión

1.1. Introducción

En el campo del desarrollo de software el advenimiento de notaciones como UML generaron un nuevo paradigma para la implementación de software, si bien esta notación no es una idea nueva, por el hecho de que tiene sus inicios en la decada de los 1990 siendo la idea principal del mismo, el modelado de sistemas de software en varios niveles de abstracción [1], además considerandose por estudios como la herramienta de notación para modelado de software dominante [2] aunque usado predominantemente de manera informal y que las actividades realcionadas al ámbito tienden a ser más fáciles en un desarrollo con un enfoque en el modelado [3] con la capacidad de además de tener beneficios dentro del entorno de desarrollo dado al nivel de expresión que se tiene en las notaciones gráficas comparadas con las formas textuales de expresión [4].

A pesar de lo mencionado, encuestas realizadas en uno de los estudios que trata a UML como una notación dominante dentro del desarrollo de software realizada años atras, muestran que desarrolladores prefieren centrar el trabajo en vastas cantidad de líneas de código, raramente mirando diagramas, y mucho menos confeccionándolos o editándolos, posibles razones para esto se pueden para este comportamiento se descubre en [5] en donde se identifican cuatro razones por las cuales los efectos positivos de UML pueden verse reducidos, aunque para el presente trabajo se puede hacer hincapié en tres de los mencionados: *la no-factibilidad de realizar ingenieria inversa en codigo heredado*, *costo de entrenamiento del equipo* y *los requerimientos no estaban distribuidos en unidades funcionales del sistema*.

En el mismo estudio, se consultó cuáles son las cuestiones de trabajar con el desarrollo orientado a modelos que lo hacen difícil de tratar o sean la razon por la cual no se modele mucho. Las razones principales fueron:

1. *Los modelos se desactualizan con respecto al código*, es decir no se encuentra un flujo de trabajo que contemple a ambas actividades tal que estas dos avancen a la par, sino que se deben realizar por separado, causando

así que el modelo quede desactualizado con frecuencia.

2. *Los modelos no son intercambiables fácilmente*, aquí se habla de la dificultad de, nuevamente, encontrar un flujo de trabajo que permita el paso de los modelos de forma más "natural".

3. *Las herramientas para modelado son pesadas*, haciendo alusión a los recursos computacionales en los cuales se debe incurrir para utilizar dichas herramientas.

4. *El código generado desde una herramienta no es de mi agrado.*

5. *Crear y modificar un modelo es lento.*

Además de la encuesta anterior, también se realizó una centrándose en el desarrollo *centrado en la codificación*, los siguientes fueron los mayores inconvenientes identificados por los desarrolladores.

1. Ver de "un pantallazo" todo el diseño es un problema, una forma de transmitir la dificultad que se tiene de brindar una visión a un alto nivel del proyecto, lo cual lleva al tercer punto de la lista.

2. Comportamiento del sistema difícil de entender.

3. La calidad del código se degrada con el tiempo.

4. Muy dificil reestructurar el proyecto.

5. Cabios de codigo pueden implementar bugs.

1.2. Desarrollo de Software Dirigido por Modelos

El desarrollo de software dirigido por modelos (MDSD, de ahora en mas, por sus siglas en inglés) como se menciona en [6] , surge a partir de la popularización de UML, el uso del mismo, sin embargo, solamente se restringía a la confección de documentación, debido a motivos ya mencionados, el acercamiento de que ofrece MSDS es enteramente diferente, la parte "dirigido" (Driven, en MDSD), enfatiza la importancia y el rol central que le da este paradigma al modelo éste ya no solamente constituye la documentación del software, sino que es considerado igual a código, este es comparable a campos de alta especialidad, dada una de las características del acercamiento, esta es que apunta a la realización de abstracciones específicas de dominio y a hacer estas abstracciones accesibles a través del modelado, ésta característica permite la automatización de la implementación de código, haciendo posible a la vez el incremento en la productividad y la mantenibilidad de los sistemas.

Para que se pueda aplicar el concepto de "modelo específico de dominio" se deben tener en cuenta tres requerimientos:

- *Lenguajes de dominio específico*, para la formulación de modelos.
- Lenguajes que puedan expresar transformaciones "modelo-código".
- *Compiladores, generadores o transformadores*, para generar el codigo ejecutable en varias plataformas.

1.3. Arquitectura Dirigida por Modelos

La arquitectura dirigida por modelos (MDA, por sus siglas en ingles) es una iniciativa introducida por el Object Management Group, con el proposito de brindar una forma estandarizada para la especificación e interoperabilidad de sistemas basado en el uso formal de modelos [7], en el núcleo de MDA se encuentran otros estándares implementados por el OMG: the Unified Modeling Language (UML), Meta Object Facility (MOF), XML Metadata Interchange (XMI) y el Common Warehouse Metamodel (CWM). Al igual que el MDSD, MDA, ubica los modelos de sistemas en el nucleo del problema de interoperabilidad lo cual hace que la implementación del sistema sea independiente de la técnología.

La OMG, promueve MDA como un *marco de trabajo arquitectónico para el desarrollo de software*, el cual esta construido alrededor de un numero de especificaciones detalladas de la misma organización que son usadas ampliamente en la comunidad de desarrolladores.

Esto puede hacer pensar que $MDA = MDSD$, lo cual sería correcto hasta cierto punto, en principio, el acercamiento de MDA es similar al de MDSD, pero difiere en detalles, por ejemplo, éste, tiende a ser mas restrictivo, enfocándose primariamente en lenguajes de modelado basados en UML [6].

1.4. Lenguaje Específico de Dominio

Un lenguaje especifico de dominio (DSL) es un lenguaje de software el cual se especializa en abarcar un problema dentro de la ingeniería en particular, las cuales son características para un dominio de aplicación, éste, se basa en abstracciones alineadas a este dominio y provee una sintaxis propicia para aplicar estas abstracciones de forma efectiva [8].

La implementación de un DSL permite mitigar ciertos aspectos que se vieron como desventaja al inicio, algunos mencionados en [9] incluyen la reutilización de código, promueve la legibilidad y el entendimiento debido al alto nivel de abstracción del mismo, permite a que usuarios con nivel bajo en programación la creación de modelos para programas siempre y cuando estos posean el conocimiento del dominio, más verificaciones en la sintáxis y semantica que un lenguaje de modelado general; aunque también se enfatizan desventajas que estos insertan en el desarrollo, curva de aprendizaje necesaria y la falta de personas letradas en el DSL, ya que es mas probable que las personas sepan como

resolver los problemas adoptando un lenguaje de propósito general el cual ya conocen.

Algunos autores definen las características deseables de un DSL [10]:

- La capacidad de describir todo el software.
- La capacidad de describir varios niveles de abstracción.
- Legibilidad y Simplicidad del lenguaje.
- Expresiones no Ambiguas.
- Soporte e Integrabilidad.

1.5. Ejemplos de Implementaciones

1.5.1. Umple

Un caso con bastante aceptación dentro del desarrollo de software llevado mediante modelos es la herramienta Umple [11], la cual tiene como objetivo mitigar varios de los inconvenientes enumerados anteriormente, con respecto a la renuencia de los equipos de trabajo para el modelado y al problema que cada desarrollador ve en un desarrollo centrado en la codificación [2], [12], esto a través de la aplicación de refactorizaciones al código lo cual da como resultado un programa equivalente al original, con el agregado de que este puede ser renderizado y editado mediante herramientas UML [13], siguiendo diferentes paradigmas dentro del desarrollo de software tales como el uso de un Lenguaje DSL (`Sección 1.4`).

1.5.2. Telosys

El análisis realizado en [14] lo muestra como una herramienta simple (que utiliza cuestiones tales como los DSL para la creación del modelo), provee la habilidad de la generación de código teniendo como base un modelo, el cual se le provee a la misma mediante una interfáz de línea de comandos, su objetivo es proveer una alternativa al clásico "Primero el UML" dentro del desarrollo, esto significa, que en vez de invertir el tiempo del desarrollador al inicio del proyecto documentando diagramas UML, teniendo como principio lo que se expuso en la `Sección 1.3` se tiene que el modelo **es** el código, es decir, que los límites que separan la documentación de la codificación se desvanecen.

Capítulo 2

Problema

Descripcion del problema que se pretende mitigar con el lenguaje

Capítulo 3

Director

Estructura del Lenguaje

3.1. Palabras Reservadas

3.2. Símbolos Especiales

3.3. Expresiones Regulares

3.4. Autómatas Finitos

3.5. Analizador Sintáctico, Gramática Libre de Contexto

3.6. Arboles de Análisis Sintáctico

3.7. Diagrama de Sintáxis

Referencias

[1] C. Michel R. V., "Empirical Studies into UML in Practice: Pitfalls and Prospects", *IEEE/ACM 9th International Workshop on Modelling in Software Engineering*, 2017. DOI: 10.1109/MiSE.2017.24.

[2] A. Aldaeej y O. Badreddin, "Towards Promoting Design and UML Modelling Practices in the Open Source Community", *IEEE/ACM 38th International Conference on Software Engineering Companion*, 2016. DOI: 10.1145/2889160.2892649.

[3] A. Forward y T. Lethbridge, "Perceptions of Software Modeling: a Survey of Software Practitioners", *5th Workshop from Code Centric to Model Centric: Evaluating the Effectiveness of MDD*, 2010.

[4] B. Rumpe, *Modelling With UML, Language, Concepts, Methods*. Aachen, Alemania: Springer, 2004. DOI: 10.1007/978-3-319-33933-7.

[5] B. Anda, K. Hansen, I. Gullesen y H. K. Thorsen, "Experience from Introducing UML-based Development in a Large Safety-Critical Project", *Empirical Software Engineering*, 2006. DOI: 10.1007/s10664-006-9020-6.

[6] T. Stahl y M. Völter, *Model-Driven Software Development, Technology, Engineering, Management*. England: John Wiley y Sons, Ltd, 2006, págs. 3-27.

[7] J. D. Poole, "Model Driven Architecture: Vision, Standards and Emerging Technologies", *ECOOP, Workshop on Metamodeling and Adaptative Object Models*, 2001.

[8] S. Sobernig, B. Hoisl y M. Strembeck, "Extracting Reusable Design Desicions for UML-Based Domain-Specific Languages: A Multi-Method Study", *Journal of Systems and Software*, 2016. DOI: 10.1016/j.jss.2015.11.037.

[9] V. C. Nguyen, X. Qafmolla y K. Richta, "Domain Specific Language Approach on Model-Driven Development of Web Services", *Department of Computer Science and Engineering, Faculty of Electrical Engineering, Czech Technical University*, vol. 11, n.º 8, 2014.

[10] M. Mazanec y M. Ondřej, "On General-Purpose Textual Modeling Languages", *Department of Computer Science, FEL, Czech Technical University*, págs. 1-12, 2012.

[11] U. Team. (2018). Umple, dirección: http://cruise.site.uottawa.ca/umple/.

[12] M. A. Garzón, H. Aljamaan, T. C. Lethbridge y O. Bradeddin, "Reverse Engineering of Object-Oriented Code into Umple using an incremental and Rule-Based Approach", 2014.

[13] T. C. Lethbridge, A. Forward y O. Badreddin, "Umplification: Refactoring to Incrementally add Abstraction to a Program", *Working Conference on Reverse Engineering*, vol. 17, 2010.

[14] L. Guérin. (2018). Telosys: the Concept of Lightweight Model for Code Generation, dirección: https://modeling-languages.com/telosys-tools-the-concept-of-lightweight-model-for-code-generation/.