Plan de Trabajo Final
Carrera Ingeniería de Sistemas
Facultad de Ciencias Exactas-UNICEN

Tema: Desarrollo de aplicaciones móviles a partir de una integración de Haxe en MDA

Alumno: Pablo Nicolás Díaz Bilotto

Directora: Liliana Favre

1. INTRODUCCIÓN

Actualmente los dispositivos móviles acompañan a los usuarios en todo momento y lugar y se prevé que serán el principal medio de acceso a Internet en los próximos años. Su uso se incrementará en todos los campos, incluyendo juegos, ocio, gestión y aplicaciones específicas para solucionar problemas en diversas áreas profesionales. Los últimos avances tecnológicos han acercado a los dispositivos móviles a las tradicionales computadoras de escritorio en cuanto a su potencia de cálculo, sin embargo, el potencial de estas tecnologías no es aprovechado aún y el desarrollo de aplicaciones de software móviles no está lo suficientemente maduro.

Desarrollar una aplicación de software para un dispositivo móvil implica adoptar y entender las características de estos dispositivos. Si bien éstos cuentan con características avanzadas como bases de datos integradas, soporte multimedia y mecanismos de comunicación y geolocalización, también se presentan importantes restricciones en cuanto al tamaño de la pantalla disponible, la utilización de memoria primaria y las bibliotecas de desarrollo disponibles [14]. Dehlinger y Dixon destacan en [4] que dos de los principales desafíos de la ingeniería de software de aplicaciones móviles son por un lado, la creación de interfaces de usuario que abarquen diferentes tipos de móviles y por otro, brindar aplicaciones reutilizables para múltiples plataformas.

La proliferación de diferentes plataformas móviles, ha forzado a los desarrolladores a definir enfoques que permitan simplificar el desarrollo de aplicaciones. Un problema actual es el desarrollo de una aplicación para múltiples plataformas móviles [15], o la

adaptación de aplicaciones o librerías existentes a las múltiples tecnologías móviles. En [7] se presenta un análisis de los conceptos claves de las plataformas Android, iPhone y QT. Wasserman analiza problemas abiertos en la ingeniería de software de aplicaciones móviles remarcando que hoy en día hay un "ángulo" móvil para todo desarrollo de software [15].

Otro desafío en desarrollos móviles es integrar variedad de información e interoperabilidad de herramientas. Model Driven Development (MDD) es considerado un enfoque promisorio para afrontarlos. Dunkel y Bruns destacan en [5] las ventajas de MDD en el contexto de aplicaciones móviles. MDD define un amplio rango de desarrollos basados en el uso de modelos como entidades de primera clase. Una realización específica de MDD, propuesta por OMG (Object Management Group), es la arquitectura MDA (Model Driven Architecture) [1] [8].

Model Driven Architecture (MDA)

La definición inicial de MDA (Model Driven Architecture) tuvo que ver con el problema de interoperabilidad de middleware en Internet. Más allá de resolver problemas de interoperabilidad, MDA ofrece otros beneficios vinculados a productividad, calidad de procesos y costos de mantenimiento. MDA propone separar la especificación de la funcionalidad del sistema de su implementación sobre una plataforma y tecnología específica y controlar la evolución del software desde modelos abstractos a implementaciones tendiendo a aumentar el grado de automatización [1]. En un proceso dirigido por modelos basado en MDA pueden distinguirse al menos las siguientes etapas:

-La construcción de un modelo, denominado CIM (*Computation Independent Model*), desde el cual se puede visualizar al sistema desde un punto de vista independiente de la computación, este modelo usualmente es llamado modelo de negocio o modelo del domino del sistema.

-La construcción de un modelo, denominado PIM (*Platform Independent Model*), que expone al sistema desde un punto de vista independiente de la plataforma.

-La transformación de un PIM en uno o más modelos relacionados a una plataforma específica (por ejemplo .NET, JEE, relacional, etc.), denominados PSM (*Platform Specific Model*).

-La transformación de modelos PSM a modelos de implementación, que algunos autores denominan ISM (*Implementation Specific Model*).

La difusión inicial de MDA enfatizaba su relación con UML como lenguaje de modelado, sin embargo hay usuarios de UML que no usan MDA y usuarios de MDA que usan otros lenguajes de modelado como específicos DSL (*Domain Specific Language*).

Una de las características esenciales de MDA es que todos los artefactos involucrados en un proceso de desarrollo se representan a partir del lenguaje de metamodelado MOF (*Meta Object Facility*) [11]. MOF define una forma común de capturar todas las construcciones de modelado e intercambio que son usadas en MDA y es la esencia de MDA al permitir que diferentes tipos de artefactos provenientes de

diferentes vendedores sean usados juntos en un mismo proyecto.

Otro concepto fundamental es el de transformaciones entre modelos. En MDA, una transformación es la especificación de mecanismos para convertir elementos de un modelo en elementos de otro modelo. El estándar propuesto por OMG para especificar transformaciones es el metamodelo QVT (*Query, View, Transformation*) [10]. QVT define tres lenguajes de transformación de modelos (QVT-Operational, QVT-Relational y QVT-Core) que conforman a MOF 2.0. Una transformación en cualquiera de estos lenguajes puede considerarse como un modelo que conforma a uno de los metamodelos especificados en el estándar. QVT esta integrado con el estándar OCL 2.0 [12]. Otros lenguajes están inspirados en los requerimientos del estándar QVT, por ejemplo el lenguaje ATL (*Atlas Transformation Language*) [6] [10]. ATL es un lenguaje híbrido que permite especificar transformaciones entre modelos de forma declarativa e imperativa. Este lenguaje es descrito mediante una sintaxis abstracta (un metamodelo MOF) y una sintaxis textual concreta. Pocas herramientas CASE basadas en MDA soportan QVT y actualmente ATL es el lenguaje de transformación más utilizado por su grado de madurez [2].

2. MOTIVACIÓN

En aplicaciones multiplataforma, los desarrolladores prefieren implementar una aplicación una vez, y desplegarla para diversas plataformas con un mínimo esfuerzo. En esta dirección, surgió Haxe, un lenguaje de programación de propósitos generales diseñado para que los desarrolladores puedan, utilizando un solo lenguaje y un conjunto de librerías, abarcar varias plataformas de manera eficaz. Actualmente, el compilador de Haxe genera código para las siguientes plataformas: JavaScript, Flash, NekoVM, PHP, C++ y C# [3]. La idea detrás de Haxe es permitir al desarrollador elegir una plataforma proveyendo un lenguaje estandarizado, una biblioteca estándar que trabaja sobre todas las plataformas que soporta y bibliotecas específicas de plataformas.

El lenguaje Haxe evoluciona con el aporte de la comunidad de usuarios a quienes brinda la posibilidad de crear aplicaciones para diversas plataformas móviles. Las herramientas CASE basadas en UML o en MDA, actualmente, no soportan procesos de ingeniería directa (*forward*) que utilicen a Haxe como plataforma. En esta propuesta, se considera un aporte valioso contar con herramientas que adhieran a MDA y/o MDD, y que soporten esta plataforma.

3. OBJETIVO Y ALCANCE DEL TRABAJO FINAL

El objetivo de este trabajo final de la carrera de Ingeniería de Sistemas es combinar desarrollos móviles en Haxe con el enfoque MDD, en particular MDA.

El primer paso en esta dirección es contar con una definición de Haxe, alineada con MDD y MDA. Se propone como objetivo definir un metamodelo MOF para esta plataforma, el cual permitirá especificar transformaciones a nivel de metamodelos y generar código a partir de un modelo. Se propone además validar la propuesta a partir de varios casos de estudio para la adaptación de aplicaciones Java a Haxe. Este metamodelo, una vez completado, podría ser utilizado además, para soportar procesos de ingeniería inversa para proyectos implementados en otras plataformas.

Para lograr los objetivos propuestos se propone el uso de una combinación de técnicas básicas de MDD y MDA como metamodelado y transformación de modelos con la plataforma Haxe. Se pretende que este enfoque facilite la adaptación de aplicaciones existentes a plataformas móviles. Se validará la propuesta bajo el proyecto Eclipse (*Eclipse Modeling Framework*) dado que algunos de sus subproyectos proporcionan herramientas y entornos de ejecución alineados con estándares de MDA [10]. Por ejemplo, EMF provee facilidades para metamodelado y un motor de ejecución para los modelos que soporta la creación de clases Java de un modelo. Otro ejemplo es

el subproyecto ATL que soporta transformaciones entre modelos. Otro subproyecto es Acceleo [8], el cual permite definir transformaciones de modelos en texto. En particular, se propone utilizar ATL como lenguaje de transformación y su conjunto de herramientas de desarrollo construidas sobre Eclipse que incluyen un motor de transformación y un ambiente de desarrollo integrado (editor, compilador y debugger).

4. CRONOGRAMA DE ACTIVIDADES

1. Estado actual del conocimiento y búsqueda bibliográfica.
   Análisis de estándares vinculados a MDA, técnicas de metamodelado y herramientas de soporte para metamodelado

2. Diseño, desarrollo, implementación y prueba del software

   a) Desarrollo del metamodelo de la plataforma Haxe

   b) Desarrollo de la transformación de Java a Haxe

3. Desarrollo de los casos de estudio

   a) Transformación de librerías de Java a Haxe.

   b) Transformación de aplicaciones completas de Java a Haxe.

5. BIBLIOGRAFÍA

[1] Marco Brambilla, Jordi Cabot, Manuel Wimmer. Model Driven Engineering in Practice. Morgan& Claypool Publishers, 2012.

[2] CASE MDA. Committed Companies and Their Products. www.omg.org/mda/committed-products.htm, 2013.

[3] Benjamin Dasnois. haXe 2 Beginner's Guide. Packt Publishing, 2011.

[4] Josh Dehlinger, Jeremy Dixon. Mobile application software engineering: Challenges and research directions. Workshop on Mobile Software Engineering, 2011.

[5] Jürgen Dunkel, Ralf Bruns. Model-driven architecture for mobile applications. In Witold Abramowicz, editor, Business Information Systems , volume 4439 of Lecture Notes in Computer Science , pages 464 477. Springer Berlin Heidelberg, 2007.

[6] Frédéric Jouault, Freddy Allilaire, Jean Bézivin, Ivan Kurtev, and Patrick Valduriez. ATL: a qvt-like transformation language. In Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications , OOPSLA '06, pages 719 720, New York, NY, USA, 2006. ACM.

[7] M. Lettner, M. Tschernuth, R. Mayrhofer, “Mobile Platform Architecture Review: Android, iPhone, QT”, in Proc. Eurocast 2011, LNCS 6928, Springer-Verlag, pp. 545-552.

[8] J. Miller, J. Mukerji. MDA Guide version 1.0.1. Technical Report, Object Management Group (OMG), 2003.

[9] Obeo. Acceleo Generator. http://www.acceleo.org, 2007.

[10] OMG. Meta Object Facility (MOF) 2.0 query/view/transformation specification, 2008.

[11] OMG. Omg Meta Object Facility (MOF) core specification, version 2.4.1, agosto 2011.

[12] OMG. Omg Object Constraint Language (OCL), version 2.3.1, 2012.

[13] Dave Steinberg, Frank Budinsky, Marcelo Paternostro, Ed Merks. EMF: Eclipse Modeling Framework . Addison-Wesley, Boston, MA, 2 edi-tion, 2009.

[14] Marcel Van Amstel, Steven Bosems, Ivan Kurtev, Luís Ferreira Pires. Performance in model transformations: experiments with atl and qvt. In Proceedings of the 4th international conference on Theory and practice of model transformations , ICMT'11, pages 198 212, Berlin, Heidelberg, 2011. Springer-Verlag.

[15] Anthony I. Wasserman. Software engineering issues for mobile application development. In Proceedings of the FSE/SDP workshop on Future of soft-ware engineering research, FoSER '10, pages 397 400, New York, NY, USA, 2010. ACM.

Firma del alumno

Avalo la presente solicitud de evaluación

Firma del director