UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Redes de Ingeniería

http://revistas.udistrital.edu.co/ojs/index.php/redes/index
http://dx.doi.org/10.14483/udistrital.jour.redes.2016.1.a03

REDES DE INGENIERÍA
[Rompiendo las barreras del conocimiento]

ARTÍCULO REPORTE DE CASO

Metodología de representación de software orientada al desarrollo ágil de aplicaciones: Un enfoque arquitectural

Software representation methodology for agile application development: An architectural approach

Alejandro Paolo Daza Corredor[1] John Freddy Parra Peña[2] Lilia Marcela Espinosa Rodríguez[3]

Para citar este artículo: Daza A.P., Parra J.F. y Espinosa L.M. (2016). Metodología de representación de software orientada al desarrollo ágil de aplicaciones: Un enfoque arquitectural. *Revista Redes de Ingeniería*. 7(1), 104-111. Doi: 10.14483/udistrital.jour.redes.2016.1.a03

Recibido: 19-Mayo-2015 / Aprobado: 9-noviembre-2015

Resumen

La generación de aplicaciones web representa la ejecución de tareas|||| repetitivas, este proceso involucra la determinación de estructuras de información, la generación de diferentes tipos de componentes y finalmente tareas de despliegue y puesta a punto de las aplicaciones. En muchas aplicaciones de este tipo los componentes generados son coincidentes entre aplicación y aplicación. Las tendencias actuales de la ingeniería de software como MDE, MDD o MDA pretenden automatizar la generación de aplicaciones sobre la base de la estructuración de un modelo que permita aplicar transformaciones con la consecución de la aplicación. Este documento pretende plasmar una base arquitectural que facilite la generación de estas aplicaciones apoyándose en la arquitectura dirigida por modelos, pero sin desconocer la existencia y actualidad de modelos arquitecturales preexistentes a las tendencias mencionadas en este resumen.

Palabras clave: arquitectura, MDA, pizarra, SOA, Web, XML.

Abstract

The generation of Web applications represents the execution of repetitive tasks, this process involves determining information structures, the generation of different types of components and finally deployment tasks and tuning applications. In many applications of this type are coincident components generated from application to application. Current trends in software engineering as MDE, MDA or MDD pretend to automate the generation of applications based on structuring a model to apply transformations to the achievement of the application. This document intends to translate an architectural foundation that facilitates the generation of these applications relying on model-driven architecture but without ignoring the existence and relevance of existing trends mentioned in this summary architectural models.

Keywords: architecture, Blackboard, MDA, SOA, Web, XML.

1. Ingeniero de Sistemas, Especialista en ingeniería de software, Estudiante del Máster en Dirección e Ingeniería de Sitios Web de la Universidad Internacional de la Rioja. apdaza@gmail.com

2. Ingeniero de Sistemas. Especialista en proyectos informáticos. Estudiante del Máster en Dirección e Ingeniería de Sitios Web de la Universidad Internacional de la Rioja. newkrux@gmail.com

3. Ingeniera de Sistemas, Especialista en ingeniería de software, Estudiante del Máster en Dirección e Ingeniería de Sitios Web de la Universidad Internacional de la Rioja. marcespinosa@gmail.com

INTRODUCCIÓN

La metodología de representación de software orientada al desarrollo ágil de aplicaciones basada en MDA usando tecnologías XML, pretende acortar los tiempos usados en el desarrollo de aplicaciones web. Gran parte del tiempo usado en el desarrollo de este tipo de aplicaciones es usado en tareas repetitivas; dicho proceso involucra la determinación de estructuras de información para la aplicación, la generación de componentes comunes a este tipo de aplicaciones, como componentes de validación y autenticación, componentes de gestión de recursos, componentes de generación de interfaces, componentes validadores, componentes de gestión de persistencia y otros más; y finalmente tareas de despliegue y puesta a punto de las aplicaciones.

Las tendencias actuales de la ingeniería de software, como la ingeniería dirigida por modelos (MDE), el desarrollo dirigido por modelos (MDD) o la arquitectura dirigida por modelos (MDA), pretenden gestionar de alguna manera las tareas de generación de este tipo de aplicaciones con la finalidad de automatizar la creación de las mismas, con base en la estructuración de un modelo que permita aplicar transformaciones y lograr la consecución de la aplicación.

En el TFM se pretende que, sobre la base de un modelo establecido mediante una representación XML, se genere el módulo completo a desarrollar de tal manera que este requiera una intervención mínima de los equipos de desarrollo para su puesta en producción, estableciendo, siguiendo el modelo, incluso los documentos de pruebas unitarias a los que se debería someter el módulo.

El alcance que se pretende en este documento es plasmar la base arquitectural que facilite la generación de estas aplicaciones, apoyándose en la arquitectura dirigida por modelos y teniendo en cuenta tecnologías XML, que permitan expresar dichos modelos.

MÉTODOS

Con base en los objetivos propuestos para el TFM es requerido determinar la base arquitectural que soporte la generación automática de software en ambiente web; a fin de determinar la misma se usa la investigación proyectiva, la cual consiste en la elaboración de una propuesta, un plan, un programa o un modelo, como solución a un problema o necesidad de tipo práctico en un área particular del conocimiento, con base en una experiencia previa, en este caso en el desarrollo de software en ambiente web.

La experiencia previa usada como base para el desarrollo de esta propuesta se encuentra dada por la participación y dirección de aplicaciones en ambiente web durante aproximadamente quince años, en los cuales se han elaborado aplicaciones en diferentes lenguajes y plataformas, donde siempre se han evidenciado las características comunes que se describen en la problemática de este documento en la tabla 1.

Aproximación dirigida por modelos al desarrollo de aplicaciones

Según Sendall y Kozaczynski, una de las formas de combatir con la complejidad del software es mediante el uso de abstracciones, la descomposición y la separación de intereses [1]. La arquitectura dirigida por modelos (MDA), la ingeniería dirigida por modelos (MDE) y el desarrollo dirigido por modelos (MDD) son aproximaciones que permiten lograr este objetivo, ya que se centran en la consecución de un modelo que luego pueda ser transformado en una aplicación; estas aproximaciones dirigidas por modelos potencian el reúso de componentes de software [2], lo cual es un objetivo base de cualquier empresa de software. Para lograr este objetivo es esencial proponer una metodología que potencie el reúso basado en un alto nivel de abstracción, es aquí donde se requiere una aproximación dirigida por modelos y conceptos, como

Tabla 1. Problemática detectada para cada fase de los proyectos.

Etapa	Descripción
Establecimiento de arquitectura tecnológica	En cada uno de los proyectos web que se han desarrollado siempre se ha establecido manualmente una arquitectura sobre aplicaciones y componentes de software básicos, los que generalmente coinciden para cada proyecto con algunas particularidades
Inicio del proceso de desarrollo	La lectura, comprensión y refinamiento de los documentos de requerimientos para cada componente de software
Proceso de desarrollo	La generación de componentes que, aunque han sido desarrollados con anterioridad, se desarrollan nuevamente por desconocimiento de su existencia o por el alto costo de adaptación de los mismos
	La creación de las estructuras de información necesarias según el modelo de información requerido para el proyecto
	La generación de interfaces de usuario de forma manual, cuando el proceso puede ser automatizado
	El afinamiento de interfaces de módulos para la integración de los componentes desarrollados
Procesos de prueba	La generación de pruebas unitarias para cada componente desarrollado
	La generación de pruebas de integración de los módulos
Procesos de documentación	La generación de documentación técnica del código implementado
	La generación de los diccionarios de datos para las estructuras de información
	La generación de otros tipos de documentos de los componentes usados
Procesos de despliegue	La adaptación en pro de la aceptación del software desarrollado
	Problemas con el manejo de versiones de los componentes desarrollados
	Problemas con el manejo de versiones de los componentes base sobre los que se despliega el software

la arquitectura dirigida por modelos MDA, así, las líneas de producción de software y los lenguajes de dominio específico cobran importancia.

Una de las promesas de MDA es facilitar la creación de modelos con el objetivo de lograr flexibilidad a largo plazo [3] en términos de: obsolescencia tecnológica, portabilidad, productividad, calidad, integración, mantenimiento, pruebas y retorno de inversión. El proceso de MDA parte de la creación de un modelo independiente de la plataforma que luego pueda convertirse en uno apropiado para una plataforma especifica; en este proceso es común que el sistema sea expresado por medio de varios modelos organizados mediante diferentes capas de abstracción.

Existen muchas propuestas de uso de UML como herramienta para la creación de tales modelos, pero UML en sí no es suficiente cuando se piensa en arquitecturas [4], ya que en parte este se encuentra atado a un paradigma en particular (orientación a objetos) y la complejidad y heterogeneidad tecnológica existente no se puede modelar solo con la orientación a objetos. Algunos de los problemas de UML para expresar arquitecturas de software son: carece de la capacidad de especificar los requerimientos de interacción entre actores, no permite expresar relaciones secuenciales, paralelas o iterativas entre casos de uso, dificulta la descomposición de sistemas distribuidos, no permite expresar correspondencia a nivel de mapeo entre elementos de diferentes vistas, y otros más.

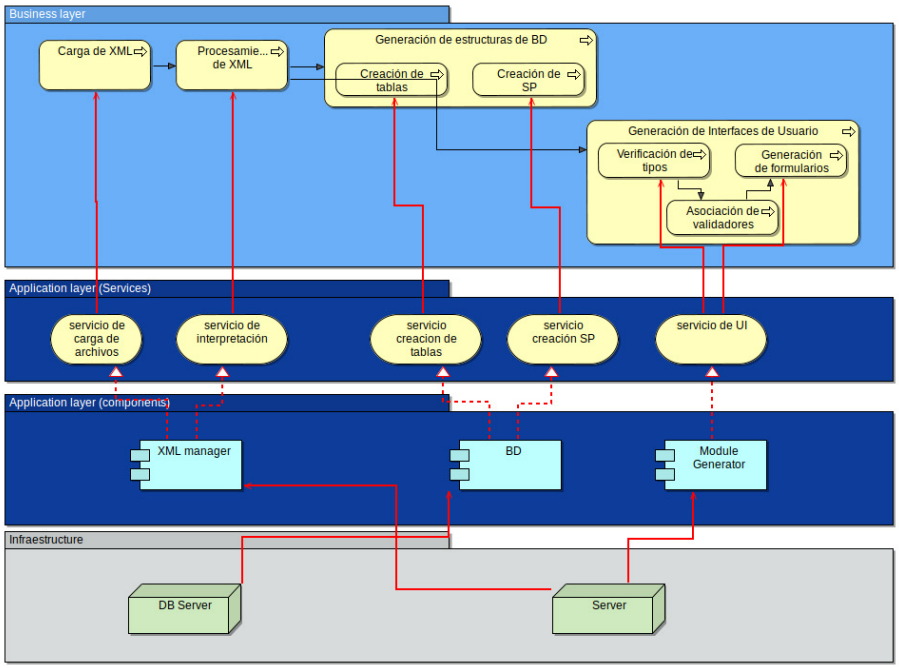

Figura 1. Propuesta arquitectural por capas y orientada a servicios para la generación inicial de módulos de software.

Por lo expuesto en el párrafo anterior, se hace necesario contemplar el estudio de elementos más allá del uso de UML, que permita representar el software desde un contexto más abstracto que el modelado orientado a objetos, para poder expresar una arquitectura que permita procesar el nivel de abstracción requerido en la generación automática de software.

No se pretende en este artículo ni en el TFM la generación de un nuevo lenguaje de modelado, ya que existen diferentes propuestas que se pueden usar con este fin, por ejemplo el framework de modelado de Eclipse (EMF) [5] o la propuesta que hacen Chu, Chang y Lu [6] [7] del uso de un dialecto XML basado en UML. La intención de este documento es presentar el modelo arquitectural que sirve de base para el TFM.

Propuestas previas realizadas

El desarrollo de la presente propuesta no es el estadio inicial de la búsqueda de una solución a esta problemática. Con anterioridad se han desarrollado aproximaciones de solución enfocándolas principalmente al proceso de desarrollo de componentes, en los cuales se ha definido una metodología de representación de componentes que cumplen con cierta estandarización de su estructura. La propuesta fue probada en unos pocos proyectos de software, pues al estar pensada solo para representar un cierto tipo de componentes, en cuanto a su estructura de información usando dialectos XML, no se orientó para otras etapas del proyecto, como la especificación de requerimientos, adaptación de módulos desarrollados previamente, documentación, pruebas y despliegue, los cuales debían seguirse haciendo con un proceso de desarrollo tradicional. Esta propuesta se representa en la figura 1.

Propuesta arquitectural

La propuesta arquitectural presentada en este documento crea una mixtura de dos estilos arquitecturales, con la finalidad de soportar el proceso de desarrollo automático de aplicaciones web, con base en un modelo independiente de la plataforma,

un estilo arquitectural centrado en datos y un estilo arquitectural orientado a servicios.

Estilos arquitectónicos centrados en datos

Los estilos arquitectónicos centrados en datos enfatizan en la integralidad de los mismos y se definen como apropiados para sistemas que se basan en el acceso y actualización de datos en grandes estructuras de información. Uno de estos estilos arquitectónicos es el estilo de pizarra.

La arquitectura de pizarra define dos componentes principales, la estructura de datos que representa el estado actual de la información y la colección de componentes que operan sobre la información. Este estilo arquitectónico tiene aplicaciones en sistemas que requieren la interpretación de procesos o en sistemas que involucran el acceso compartido a datos, con agentes débilmente acoplados o en sistemas organizados como colecciones de herramientas en torno a un repositorio común [8].

Estilo arquitectónico orientado a servicios

El estilo arquitectónico orientado a servicios se fundamenta en la expansión de los Web Services basados en XML y el protocolo SOAP. El centro de esta arquitectura está definida por componentes independientes que se comunican mediante mensajes y redefine los estilos arquitectónicos orientados a objetos y componentes.

La orientación a servicios utiliza estándares Web para los formatos de datos y los protocolos de aplicación. Esto permite un mayor grado para la interoperabilidad, ya que es posible implementar tratamiento de XML y SOAP en casi cualquier lenguaje y plataforma [8].

Elección arquitectónica

Dadas las ventajas de ambos estilos arquitecturales se plantea una mixtura de estilos que permitan aprovechar las ventajas de cada uno de ellos haciendo uso de dialectos XML, para representar software desde sus requerimientos hasta despliegue, que permita solventar la problemática actual descrita en el documento.

Arquitecturalmente se plantea el uso de un estilo arquitectural de pizarra, ya que toda la interoperabilidad estaría centrada en el manejo de la información generada en los procesos de especificación del modelo de software y trabajaría sobre un repositorio central de componentes pre-establecidos y agentes generadores de transformaciones de los modelos independientes de plataforma especificados.

Al hacer referencia a la pizarra se habla de componentes independientes que usan la información de las pizarras, aquí se plantea el uso de la orientación a servicios para comunicar dichos componentes de forma escalable e integrable y se cruzan directamente con los dialectos XML que permiten representar el software.

Al realizar la implementación de estos dos estilos arquitectónicos y los estándares seleccionados se permite que, independientemente de las necesidades particulares de cada proyecto de software, se automatice su proceso de desarrollo sobre la base del conocimiento que se pueda manejar en la pizarra y la integración que permite la orientación a servicios. Este modelo arquitectural permite la evolución, escalabilidad, extensibilidad y otras características deseadas y requeridas en los proyectos de software, soportando las ventajas que nos ofrece la aproximación dirigida por modelos en la construcción de software.

En la generación del estilo arquitectural de pizarra se generan los repositorios de componentes preexistentes, que permiten la generación de transformaciones sobre los modelos establecidos, también se manejarían componentes de repositorios de modelos asociados a proyectos; es en este componente donde cobra vital importancia el manejo de la

pizarra, ya que solo interesa el manejo del estado actual de la representación del software y los agentes relacionados con la pizarra solo trabajan con este estado para generar la transformación. En la figura 2 se ve reflejado este planteamiento.

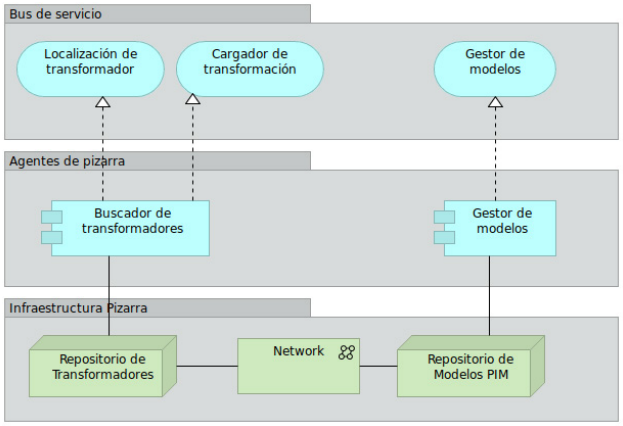

Figura 2. Punto de vista introductorio para la pizarra.

La responsabilidad de SOA debe ser permitir la integración de todos los servicios que se monten sobre los agentes que se integren con la pizarra, y permitir la independencia tecnológica sobre los componentes que intervengan en el proceso de generación de software.

Para la validación del modelo arquitectural se generaron prototipos de los componentes de "Buscador de transformadores" y "Gestor de modelos", de igual forma, se aplicaron a los siguientes modelos como se puede ver en la tabla 2.

Las dos transformaciones se integraron en un proyecto en el que no se presentaron errores para la integración de las mismas.

CONCLUSIONES

Las transformaciones creadas en el framework sobre el que se desarrollaron las pruebas se orientaron a la generación de código en PHP, que serían equivalentes a las que se desarrollaran para cualquier elección de lenguajes de programación o plataformas tecnológicas. Esta característica es soportada por el modelo arquitectural propuesto, dada su escalabilidad funcional soportada en las pizarras y la orientación a servicios, así como la facilidad de la generación de transformaciones que se pueden hacer con las tecnologías XML.

Tabla 2. Modelos de prueba de transformación.

```
<?xml version="1.0" encoding="UTF-8"?>
<component type="entidad" name="compromiso">
  <location folder="aplicacion" host="local"/>
  <attributes>
    <variable name="id" type="int"/>
    <variable name="responsable" type="responsable"/>
    <variable name="fecha_limite" type="date"/>
    <variable name="descripcion" type="text"/>
  </attributes>
</component>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<component type="datos" name="data">
  <location folder="datos" host="local"/>
  <attributes>
    <variable name="host" type="text"/>
    <variable name="user" type="text"/>
    <variable name="password" type="text"/>
    <variable name="db" type="text"/>
  </attributes>
```

```
  <methods>
    ...
    <method name="consultar">
      <param name="cadena" type="text"/>
      <body>
        <assign var="result">
          <call name="query" type="native">
            <param value="cadena"/>
          </call>
        </assign>
        <return var="result"/>
      </body>
    </method>
    ...
  </methods>
</component>
```


ISSN: 2248–762X • Vol. 7, No. 1 (ene-jun 2016). pp. 104-111

Tabla 3. Código resultante de la transformación.

```
Class CCompromiso {
   var $id = null;
   var $responsable = null;
   var $fecha_limite = null;
   var $descripcion = null;

   function setId($val) {
      $this->id=$val;
   }
   function setResponsable($val){
      $this->responsable = $val;
   }
   function setFechaLimite($val) {
      $this->fecha_limite=$val;
   }
   function setDescripcion($val) {
      $this->descripcion=$val;
   }
   function getId() {
      return $this->id;
   }
   function getFechaLimite() {
      return $this->fecha_limite;
   }
   function getResponsable(){
      return $this->responsable;
   }
   function getDescripcion() {
      return $this->descripcion;
   }
}

Class CData {

   var $host = null;
   var $usuario = null;
   var $password = null;
   var $database = null;
   var $log = null;

   …
   function ejecutarConsulta($sql) {
      $result = mysql_query($sql);
       return $result;
   }
   …
}
```

REFERENCIAS

[1] S. Sendall, W. Kozaczynski. *Model Transformation – the Heart and Soul of Model-Driven Software Development*. INFOSCIENCE, S.f. [en línea]. Consultado el 10 de noviembre del 2014, disponible en: http://infoscience.epfl.ch/record/52559/files/IC_TECH_REPORT_200352.pdf.

[2] J.B. Quintero, R. Anaya. "MDA y el papel de los modelos en el proceso de desarrollo de software", *Revista EIA A, ISSN 1794-1237 8*, 131-146, 2007.

[3] F. Truyen. *The Fast Guide to Model Driven Architecture The Basics of Model Driven Architecture (MDA)*. Cephas Consulting Corp, S.f. [en línea]. Consultado el 20 de noviembre de 2014, disponible en: http://www.omg.org/mda/mda_files/Cephas_MDA_Fast_Guide.pdf

[4] C. Reynoso, N. Kicillof. *Lenguajes de Descripción de Arquitectura (ADL)*. Buenos Aires: Universidad de Buenos Aires. [en línea]. Consultado el 10 de noviembre de 2014, disponible en: http://carlosreynoso.com.ar/archivos/arquitectura/ADL.PDF

[5] C. Vicente, D. Alonso. *Herramientas Eclipse para el Desarrollo de Software Dirigido por Modelos. División de Sistemas e Ingeniería Electrónica (DSIE), Cartagena:* Universidad Politécnica de Cartagena. [en línea]. Consultado el 20 de noviembre de 2014, disponible en: http://repositorio.bib.upct.es/dspace/bitstream/10317/1216/1/hed.pdf.

[6] W. C Chu, C. Chang, W. C. Lu. *Model-based Object-oriented Requirement Engineering and its Support to Software Documents Integration*, S.f. [en línea]. Consultado el 20 de septiembre del 2014, disponible en: http://ieeexplore.ieee.org/

[7] WC. Chu. *Improving Software Evolution and Maintenance by Using Design Patterns and an XML-based Unified Model*. Feng Chia University, e-Thesis, 2003.

[8] C. Reynoso. *Estilos y patrones en arquitectura de software*, Buenos Aires: Universidad de Buenos

Aires, S.f. [en línea]. Consultado el 20 de noviembre del 2014, disponible en: http://carlosreynoso.com.ar/wp-content/plugins/download-monitor/download.php?id=155

[9] J. Siegel. *Using OMG's Model Driven Architecture (MDA) to Integrate Web Services. Object Management Group White Paper*, 2002 [en línea]. Consultado el, disponible en: http://secure.omg.org/mda/mda_files/MDA-WS-integrate-WP.pdf

[10] S. Sendall, & W. Kozaczynski. *Model transformation: The heart and soul of model-driven software development*, 2003 [en línea]. Consultado el 10 de noviembre del 2014, disponible en: http://ieeexplore.ieee.org/

[11] M. Sánchez, A. Fermoso, & L. Joyanes. *From the platform independent model (pim) to the final code model (fcm) according to the model driven architecture (mda), 2003–2006*, 2005 [en línea]. Consultado el 10 de noviembre del 2014, disponible en: http://www.researchgate.net/publication/220969458_From_the_platform_independent_model_(PIM)_to_the_final_code_model_(FCM)_according_to_the_model_driven_architecture_(MDA)

[12] L. Quin. XML Essentials. Retrieved April, 13–15, 2010 [en línea]. Consultado el 8 de noviembre del 2014, disponible en: http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:XML+Essentials#5

[13] J. Poole. *Model-driven architecture: Vision, standards and emerging technologies. on Metamodeling and Adaptive Object Models* (April), 1–15, 2001 [en línea]. Consultado el 8 de noviembre del 2014, disponible en: http://www.adaptiveobjectmodel.com/ECOOP2001/submissions/Model-Driven_Architecture.pdf

[14] R. Picek, & V. Strahonja. *Model Driven Development-future or failure of software development*. IIS, 2007 [en línea]. Consultado el 11 de noviembre del 2014, disponible en: http://old.foi.hr/CMS_home/znan_strucni_rad/konferencije/IIS/2007/papers/T12_01.pdf

[15] R. Marvie. *A transformation composition framework for model driven engineering*, (November), 2004 [en línea]. Consultado el 7 de noviembre del 2014, disponible en: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.111.2065