Revisión de la complejidad en la creación de interfaces de usuario a partir de modelos

Autores
Víctor Hugo Mercado Ramos
Julian Zapata Arbeláez

Asesor:
Jesús Andrés Hincapié Londoño

Programa:
Ingeniería de software

Facultad:
Ingenierías

Revisión de la complejidad en la creación de interfaces de usuario a partir de modelos

Víctor Mercado[1], Julián Zapata[2]

[1]Estudiante Especialización en Ingeniería de Software Universidad de Medellín, *victorh1mercado@gmail.com*
[2]Estudiante Especialización en Ingeniería de Software Universidad de Medellín, *jjzapata2004@gmail.com*

Resumen

El Desarrollo Dirigido por Modelos (MDD por sus siglas en inglés) es un nuevo paradigma para el desarrollo de software que facilita, entre otros aspectos, la mejora en el entendimiento real de un problema, lo cual se logra gracias al nivel de abstracción que ofrece el trabajo con modelos. Además de ésta, existen otras ventajas del uso de esta metodología, sin embargo existen también desventajas. Al utilizar los modelos para crear interfaces de usuario se puede reducir, en forma notoria, el tiempo dedicado al desarrollo de las mismas, sin embargo, en ciertas ocasiones, esta generación se realiza a través de plantillas predeterminadas, que pueden no cubrir todos los requerimientos del cliente o que, por el hecho de provenir de plantillas, pueden ser muy generales y con pocas opciones de personalización. El presente artículo hace una revisión de diferentes artículos en los cuáles se aborda el tema del uso de MDD en la creación de interfaces de usuario y en cómo esto puede tener ciertas limitantes debido a la generación de interfaces predeterminadas que no son tan flexibles y que para satisfacer las necesidades del proyecto requieren la continua intervención tanto en el modelo como en el código en sí.

Palabras clave: Desarrollo Orientado a Modelos, Creación de interfaces de usuario, Desarrollo de software, Desarrollo de interfaces de usuario, uso de plantillas.

1. Introducción

La principal promesa de las empresas desarrolladoras de software es el cumplimiento de las expectativas del cliente, sin embargo, es algo natural que en el ciclo de vida de un proyecto de desarrollo, estas expectativas cambien, dándose esto por diversas situaciones tales como la maduración de la idea, el replanteamiento de objetivos o la falta de claridad en los alcances de los mismos al inicio del proyecto. También es natural que en las etapas tempranas del proyecto, se recurra al uso de modelos que permitan la abstracción del problema a resolver para tener un mejor entendimiento de éste.

El Desarrollo Dirigido por Modelos propone, como su nombre lo indica, un proceso de desarrollo basado en la creación y transformación de modelos, fundamentado en la abstracción, automatización y la estandarización (Bernardo & Anaya, 2007). Los modelos son transformados en aplicaciones funcionales mediante el uso de herramientas. El uso de esta metodología tiene ventajas tales como el mejor entendimiento del problema en sí debido al alto grado de abstracción que poseen los modelos o el ahorro de tiempo en la construcción de las interfaces, sin embargo, al aplicar estas transformaciones, existe la posibilidad de obtener interfaces muy similares las unas a las otras y que posiblemente difieran con el estilo corporativo del cliente, o interfaces que no satisfacen todos los requisitos planteados inicialmente debido a que éstos no estuviesen cubiertos por las opciones que dispone la herramienta generadora de la transformación. Para solucionar este inconveniente, es necesario realizar modificaciones al modelo desde el código mismo, lo cual implica que el desarrollador posea conocimientos adicionales en transformaciones de modelo, ya que éstas pueden llegar a ser complejas de expresar y de implementar debido a la dificultad de comprender el código generado por la herramienta que realiza la transformación. Adicionalmente, una interfaz que ha sido modificada manualmente en su código y que por algún

cambio solicitado por el cliente debe ser re-generada basada en un nuevo modelo, todos los cambios realizados de forma manual se perderán. Todos estos cambios a los cuales se somete el modelo original, requieren un seguimiento especial, por lo cual es imprescindible que la manera en la que se hagan las modificaciones pueda ser trazable.

En el desarrollo de este artículo se revisará actualmente cómo se está afrontando la situación ya descrita.

2. Método

Para llevar a cabo esta investigación fue necesario detenerse a delimitar el problema, por tanto se analizaron varias perspectivas respecto a la temática de las transformaciones de modelo y se definió que el foco de atención sería la creación de interfaces de usuario partiendo del uso de modelos. Al investigar sobre el tema (Aquino N., 2009 y Trujillo S., 2009) se encontró que existe una problemática en esta área relacionada con la poca flexibilidad, que tienen las herramientas transformadoras, para generar interfaces diferentes, además de la falta de disponibilidad de opciones que cubran la totalidad de los requisitos del proyecto.

La búsqueda de la información se realizó en bases de datos científicas, las cuales contienen un sin número de artículos y se utilizaron palabras claves que permitieron delimitar la búsqueda y encontrar de manera más rápida la información necesitada. Seguidamente se procedió a leer los resúmenes, filtrando aquellos que se consideraban adecuados para el tema de investigación y haciendo un segundo filtro basado en las conclusiones para finalmente determinar si ese artículo sería o no de utilidad para la investigación.

2.1 Preguntas de investigación

¿Cómo se puede dar más flexibilidad a las herramientas transformadoras de modelos para evitar que las interfaces creadas sean muy similares las unas a las otras y que, a su vez, cumplan todas las especificaciones del proyecto?

¿Cómo es posible hacer seguimiento a los cambios que se realizan sobre los modelos para que estos se ajusten a las necesidades del cliente?

2.2 Criterios de inclusión y exclusión

El primer filtro realizado fue la búsqueda de artículos que abordaran el tema de las problemáticas detectadas en la transformación de modelos en MDD. Sobre este tema se encuentra una amplia variedad de artículos. Posteriormente se recurrió a la formulación de la pregunta de investigación, lo cual delimitó drásticamente la cantidad de bibliografía, ya que es un problema muy específico y que no ha sido abordado por muchos autores. Los artículos incluidos fueron aquellos cuyo tema central era la generación de interfaces de usuario a través de MDD y aquellos que hacían referencia a la trazabilidad de los cambios realizados sobre los modelos que, además de abordar el tema, ofrecían alternativas para la solución del mismo de una manera concreta. Los artículos excluidos fueron aquellos que tocaban el tema de la creación de interfaces de manera muy superficial o que simplemente no tocaban el tema desde la perspectiva planteada en la pregunta de investigación. También se excluyeron los artículos que dejaban el tema en discusión y que no planteaban una solución específica.

2.3 Trabajos seleccionados

Tabla 1. Trabajos seleccionados

ID	Título	Autores	Fuente
1	Adding Flexibility in the Model-Driven Engineering of User Interfaces	(Aquino, 2009)	ACM
2	Feature-Oriented Refinement of Models, Metamodels and Model Transformations	(Trujillo, Zubizarreta, Mendialdua, & de Sosa, 2009)	ACM
3	Maintaining Invariant Traceability through Bidirectional Transformations	(Yu et al., 2012)	ACM
4	Model-Driven Engineering of User Interfaces: Promises, Successes, Failures, and Challenges	(Vanderdonckt, 2008)	ACM
5	Diseño de reglas de adaptación y transformación para interfaces de usuario	(Montero, Inform, & Mancha, 2010)	Universidad Nacional de Colombia

ID	Título	Autores	Fuente
6	Generative Pattern-Based Design of User Interfaces	(Vanderdonckt & Simarro, 2010)	ACM

3. Evaluación de los trabajos seleccionados

Para realizar la evaluación de los artículos seleccionados, se hace una valoración cuantitativa de acuerdo a los criterios enunciados en la sección 3.1. La escala usada para calificarlos es la siguiente:

3: Cumple totalmente con el criterio.
2: Cumple parcialmente con el criterio.
1: No cumple con el criterio.

3.1 Criterios de evaluación

Los siguientes criterios nos ayudan a evaluar los artículos seleccionados para, posteriormente realizar la discusión final.

Criterio 1 (C1). Las alternativas planteadas para solucionar el problema están construidas sobre bases de conocimiento sólido y de manera coherente.
Criterio 2 (C2). Los ejemplos que se dan para describir la problemática son claros y aplicables.
Criterio 3 (C3). La solución que se plantea puede ser aplicada de manera independiente a la plataforma tecnológica en la cual se construya la solución.
Criterio 4 (C4). Se plantea de manera clara la problemática que conlleva el desarrollo de interfaces de usuario mediante el uso de generación automática de código a partir de modelos.
Criterio 5 (C5). La solución presentada ha sido probada y verificada, no se encuentra aún en estudio.

3.2 Resultado de la evaluación

Según los criterios ya mencionados, a continuación se presenta una tabla resumen de los resultados de la evaluación, en la cual se asigna a cada artículo una calificación por cada criterio.

Tabla 2. Resumen de la evaluación

		Criterios de evaluación				
		Criterio 1	Criterio 2	Criterio 3	Criterio 4	Criterio 5
Trabajos	ID1	3	3	3	3	2
	ID2	3	3	3	2	2
	ID3	3	3	1	2	3
	ID4	3	3	3	3	3
	ID5	3	2	3	3	3
	ID6	3	2	1	2	3

En ***ID1*** vemos que la problemática que se plantea es propiamente la generación de interfaces de usuario a través de plantillas, ofreciendo alternativas que contrarresten la problemática de la similitud entre las interfaces generadas y la opción de agregar diferentes funcionalidades a cada interfaz. Para ello se introduce el concepto de Perfil de Transformación. Un perfil de transformación tiene dos componentes, el primero de ellos son las *Plantillas de Transformación*, las cuales son un conjunto de parámetros en los que se recopilan los elementos gráficos y de estilo que componen la interfaz. El segundo componente se puede denominar un *Modelo de Correspondencias* el cual comprende una serie de relaciones entre modelos, desde un modelo inicial o base, hasta un modelo objetivo, abarcando los elementos personalizados que pretenden sumar funcionalidad a las interfaces.

En el artículo ***ID2*** indica como para hacer seguimiento a las variaciones en las transformaciones de los modelos, es necesario tener en cuenta tres conceptos básicos: modelos, metamodelos y transformaciones de modelos. Los modelos son la representación del problema al cual se requiere dar solución, los metamodelos

son descripciones de las características del modelo y las transformaciones de modelos se refieren al proceso cuyo artefacto de entrada es un modelo y el de salida es el modelo con las variaciones que se requieren para personalizar la funcionalidad de la solución. Una de las herramientas propuestas para realizar las transformaciones es MOFScript (la cual se puede descargar como un plugín de eclipse). Las transformaciones que realizan estas herramientas están basadas en reglas. También se introduce el término *refinamiento de los modelos*, el cual permite extender un modelo adicionando elementos, con el fin de facilitar su personalización para diferentes necesidades. Esta característica hace posible la construcción de modelos base, los cuales son generalizaciones que poseen elementos comunes y que posteriormente se usan para obtener otros modelos con diferentes variaciones.

En el artículo ***ID3*** se aborda el tema de las variaciones del modelo para que éste se pueda ajustar más a las necesidades específicas, sin embargo este artículo se centra más en la importancia de tener una trazabilidad cada vez que se realizan estos cambios. Los cambios los pueden realizar los desarrolladores a través de modificaciones de manera paralela tanto en el código como en el modelo como tal. Para que siempre exista trazabilidad en los cambios que se realizan, cada vez que se hagan modificaciones en el código éstas deben ser reflejadas en el modelo. De la misma forma, cuando se realicen cambios al modelo, en el código se deben poder observar tales cambios, a esto nos referiremos como transformaciones bidireccionales. Sin embargo, es posible que en algunas ocasiones no haya una correspondencia del 100% entre los cambios que se realizan sobre el uno o sobre el otro. También se evidencia en este artículo cuáles son las consecuencias de la falta de correspondencia entre las plantillas generadas y las modificaciones realizadas por el usuario y propone un framework de dos capas que combina los cambios a través de transformaciones bidireccionales. La primer capa, denominada código – modelo, sincroniza los cambios estructurales entre el modelo y la plantilla usando EMF (Eclipse Modeling Framework), mientras que la segunda capa, código – código, sincroniza los cambios de funcionalidad entre el código de la plantilla y el código del modelo. El framework advierte cuando hay

diferencias significativas entre el modelo y el código. El proceso para la creación de estos modelos parte de la generación automática de código. Una vez que esto ocurre, el desarrollador toma el código y lo manipula para personalizar y añadir las funcionalidades que requiera. Este cambio que se introduce modifica el modelo, por tal motivo es necesario regenerar la plantilla original.

El artículo ***ID4*** se centra en la generación de interfaces de usuario a través de MDD y en cómo estas se pueden adaptar a los cambios que surgen cuando el usuario lo requiera. Este artículo describe desde el principio todos los conceptos de MDD de manera simple. Hace también referencia a los diferentes niveles que componen el MDD. El primer nivel es Tarea y dominio del modelo, en ella se especifica una tarea de usuario final, que es la que da origen al modelo. El segundo nivel se denomina Interfaz de Usuario Abstracta, en la cual se tienen las interfaces de manera independiente de la plataforma tecnológica que se use. En el tercer nivel se tienen las Interfaces de Usuario Concretas, que a diferencia del nivel dos, ya tiene definida una plataforma tecnológica. Por último se tiene la interfaz de usuario final, en este nivel ya se encuentra la interfaz producida a partir de los niveles anteriores. En este artículo también se define el concepto de modelo, el cual se aborda de manera diferente a como se hace en ID2, puesto que se enfoca en los diferentes tipos de modelo que se aplican en el desarrollo de interfaces. Posteriormente se habla de metodología y de herramientas.

En el artículo ***ID5*** se plantea un enfoque diferente al papel y la importancia que puede jugar el MDD sobre el diseño de interfaces debido a que, adicionalmente a las dificultades ya planteadas, se enfrenta el hecho de interfaces que van orientadas a usuarios finales con muy bajos conocimientos técnicos, debiendo ser estas muy intuitivas. También se adiciona otro hecho: la cantidad de dispositivos desde los cuáles actualmente se acceden a aplicativos, tales como tablets, smartphones, etc. en los que la interfaz debe adaptarse a las características específicas del dispositivo. Cuando se trabaja con modelos, se parte de un modelo inicial muy abstracto, pero a medida que se requiere, dicho modelo sufre una serie de transformaciones que lo llevan a un modelo más concreto y, finalmente, al

código final. Las transformaciones que va sufriendo el modelo inicial, deben seguir un conjunto de reglas particulares, así es como acá se define el concepto de Reglas de Adaptación. Las Reglas de Adaptación se componen de un contexto en el cuál son válidas, un evento que las dispara según el contexto, los datos a los cuales la regla debe acceder y la transformación resultante de la aplicación de la regla. Posteriormente el artículo introduce a T:XML como una herramienta que permite especificar las reglas de adaptación.

El ***ID6*** plantea como Idea central la implementación de un nuevo método para el desarrollo de interfaces de usuario basada en patrones generativos, este método consiste en 4 ejes los cuales se apoyan en la administración de los patrones y en el uso principalmente de un patrón de diseño llamado Markup Language (PLML), el cual nació en el CHI'2003 (Fincher et al., 2003), este patrón surge del análisis generado a los patrones de diseño de interfaces que en ese momento existían y lo que busca es dar solución a los problemas que estos presentaban al momento de hacer transformaciones de modelos, problemas tales como ambigüedad en las interfaces o incoherencias en las mismas, pero al ser consciente de esta problemática el nuevo patrón busca como disminuir estas falencias por medio de unas reglas para el uso del mismo,. También hace parte de los ejes de este método una aplicación llamada IDEALXML, este es un software desarrollado en Java con el objetivo de apoyar el uso de patrones.

4. Discusión

El modelado de los sistemas de software es esencial a la hora de enfrentar sistemas demasiados complejos, permitiendo al analista capturar de una forma mucho más entendible, aspectos relevantes del sistema llevándolo hasta un punto de abstracción lo suficientemente adecuado.

El Desarrollo Dirigido por Modelos apunta de manera directa a la mejora de la productividad en un proyecto software, mediante la automatización de tareas repetitivas, por ello en el artículo ***ID1*** se presentan herramientas que permiten

generar rápidamente interfaces de usuario, sin embargo, existe la posibilidad de que tales interfaces sean muy similares las unas a las otras o que no satisfagan por completo las necesidades del proyecto, para ello se introduce el concepto de Perfil de Transformación. El desarrollo del tema se vuelve demasiado técnico, la parte conceptual se relega demasiado, esto hace que el artículo, pese a su brevedad, no sea fácil de leer ni de comprender ya que muchos de los conceptos que se mencionan son explicados de forma muy ligera, obligando al lector a complementar información adicional para poder tener una mejor comprensión de las ideas planteadas. Al final del artículo se especifica como característica principal del método propuesto, el uso de plantillas para separar algunas características especiales y realizar sobre ellas las transformaciones, hecho que, según los autores, les da ventaja sobre otros estudios realizados en esta área de investigación.

En ***ID2*** el contenido del artículo se enfoca más en la importancia de llevar un control sobre las variaciones que se realizan sobre el modelo, mientras que el tema de la generación de interfaces no se trata de una manera directa. Pese a esto, se considera que el artículo ofrece un gran aporte al tema de interés, ya que se puede presentar como un complemento de cualquier otro de los artículos seleccionados, puesto que solo en el artículo ID3, se referencia de manera directa que pasa cuando se requiere dar un paso atrás y regresar a un modelo previo. Es un trabajo conciso que involucra al lector a través de un ejemplo que evoluciona con el desarrollo del tema con el fin de ir introduciendo los conceptos que se van generando. Por la forma en que se explican los conceptos y por los ejemplos que se plantean, este artículo es de fácil entendimiento, incluso para el lector que no posea fuertes conocimientos en programación. La estructura sobre la cual se desarrolla el tema es coherente, a diferencia de los otros artículos, no se hace tanto uso de las referencias que, en algunos casos, pueden distraer y alejar al lector un poco del tema. La única parte en la cual se deja una sensación de incompletitud es en las conclusiones, ya que no hacen un buen resumen de todo lo leído en el mismo.

En ***ID3*** se indica la importancia de realizar modificaciones paralelas, tanto en el código que se genera como en los modelos propiamente, esto con el fin de dar una mayor flexibilidad a las interfaces que se generan para que se puedan personalizar según sea la necesidad. El artículo se enfoca mucho a cómo algunas características que se van construyendo requieren, en algún momento, regresar a un paso anterior, por tal motivo la importancia de tener una trazabilidad de los cambios que se van realizando ya que los modelos y las plantillas van evolucionando hasta llegar a un punto en el cual pueden diferir mucho del modelo original. A diferencia del artículo ID2, en este artículo se muestra la necesidad de mantener la trazabilidad de los cambios que se realizan, es por el hecho de realizar tanto modificaciones al modelo como al código, de manera paralela. Entre las diferentes opciones que se han observado, este artículo en particular muestra un tema con un nivel más alto de complejidad, ya que su propuesta es no solo la transformación del modelo como tal, sino la de modificar el código. Este tipo de afectaciones puede tener serias consecuencias mientras más complejo sea la interfaz que se desee diseñar, lo cual puede llevar al lector a cuestionarse sobre su utilidad, ya que al revisar otras alternativas se han observado herramientas potentes en las cuáles no es necesario intervenir el código de una manera tan directa o específica.

En ***ID4*** el planteamiento que se da al artículo en primera instancia puede parecer muy general ya que introduce demasiados conceptos aunque, vale la pena resaltar, que son explicados de una forma sencilla y completa. Se puede observar también que este artículo se puede dirigir a un público con conocimiento en desarrollo pero con poco o ningún conocimiento de trabajo con modelos, puesto que la estructura en la cual está construido da pie a una evolución del tema partiendo de algo muy general como lo es el concepto de modelo y llegando hasta introducir herramientas propiamente empleadas en MDD. Este artículo no se enfoca en una herramienta o método particular, más bien ofrece un compendio de alternativas en las cuáles no profundiza sino que deja los elementos básicos para su comprensión. La profundización en estos temas queda como tarea del lector con la ayuda de una muy completa referencia bibliográfica especificada para tal

fin. También se pueden observar elementos que se describen en otros artículos de la presente revisión, tales como líneas base, reglas de asignación, etc. que permiten al lector familiarizarse para realizar comparaciones y emitir juicios. Todas estas características hacen que de los artículos revisados, éste sea el de mayor completitud y su enfoque se dirige más a un lector inexperto que desea conocer el tema sin profundizar demasiado en él.

Tras la revisión de varios artículos, se puede ir determinando que cada una de las propuestas aporta a características específicas, pero dejan de lado otras que pueden ser significativas. Realmente de lo que se alcanza a revisar en el presente artículo podría tener un enfoque más de complemento que de comparación, puesto que cada propuesta ataca un frente diferente. Entra entonces un cuestionamiento: ¿en qué nivel de madurez se encuentra el tema de generación de interfaces de usuario con el uso de MDD? Se ha mencionado de manera reiterativa la importancia del uso de MDD al igual que los riesgos o dificultades encontradas, pero finalmente no se ha encontrado una solución definitiva a todas ellas. El tema del artículo ***ID5***, es un perfecto complemento de los artículos ID1, ID2 e ID3, puesto que su planteamiento es el diseño de reglas de adaptación. El enfoque más poderoso de este tema es la posibilidad de reutilización, siendo una de las características más valiosas puesto que apunta de manera directa al mejoramiento de la calidad. El contenido y estructura en la que se escribe este artículo puede que no deje una completa sensación de satisfacción; hay algunas partes en las cuáles el lector puede sentirse un poco perdido por la falta de información y la claridad de los ejemplos deja un poco que desear, dirigiendo el artículo a un tipo de público un poco más avanzado u obligándolo a profundizar.

El ***ID6***, es un artículo que propone seguir una metodología basada en el manejo de patrones para poder realizar transformaciones de interfaces, utilizando una aplicación que cuenta con un potente número de líneas de códigos, los cuales corresponden a patrones almacenados en la misma. De este artículo se puede decir que es muy completo respecto a la solución que propone, ya que en él están especificados todos los componentes que dieron pie a la fabricación de esa

aplicación, se habla mucho de los patrones y de cómo estos aportan al a solución del problema de las transformaciones, sin omitir los problemas a los cuales se expone con su uso. Este artículo pretende también que el lector tenga un mejor entendimiento respecto al funcionamiento completo de la herramienta, mediante ejemplos claros que ayudan al lector a ponerse en contexto con lo que se está abordando. Se puede decir que este articulo está enfocado para lectores de nivel medio en cuanto a conocimiento de MDD, puesto que se toman conceptos que se aclaran solo si éstos están implícitos en la solución propuesta, mientras que si los conceptos no son tan cercanos a la solución, se tratan de manera muy superficial.

5. Conclusiones

El modelado de sistemas es una poderosa herramienta para la comprensión y diseño de solución de un problema determinado. La utilidad de esta herramienta es la capacidad de llevar un problema complejo a una representación abstracta del mismo, descomponiéndolo en subsistemas más sencillos de afrontar. Por ello, poder obtener interfaces de usuario directamente a partir de tales modelos, significa no sólo un incremento en la productividad, sino una mayor calidad en el producto software, puesto que al proceder del modelo supone el cubrimiento de toda la funcionalidad esperada por el cliente.

La generación de código a partir de modelos es una forma rápida de obtener interfaces de usuario, con muchas ventajas. Entre ellas podemos contar la inclusión de elementos de calidad dentro de las plantillas de generación de modelo y la automatización de tareas repetitivas, sin embargo, según lo que se ha planteado a lo largo del presente artículo, es muy difícil desligar la intervención de tareas humanas durante todo el proceso de desarrollo.

Durante el ciclo de vida de un proyecto software, es inevitable que surjan cambios constantemente para ajustarse a las necesidades del cliente. En muchas ocasiones estos cambios deben ser reversibles, bien sea porque la necesidad específica del cliente cambia, por la poca viabilidad de la solución que plantea el

cambio, etc. En cualquiera de estos casos es de suma importancia poder reversar el cambio realizado, por ello la importancia de mantener una trazabilidad de todos los cambios que se realizan en el transcurso del desarrollo del proyecto.

La generación completamente automática de interfaces de usuario a partir de modelos aún tiene retos importantes que afrontar, sin embargo las alternativas que se ven ofreciendo, bien sea porque están plenamente funcionales o porque están en proceso de desarrollo, son bien interesantes. El concepto como tal de MDD es muy atractivo, por todas las ventajas que ofrece y, superando estos retos pendientes, puede significar un avance importante para el desarrollo de proyectos de manera rápida sin desmejorar aspectos de calidad.

Finalmente se puede decir que son varias las opciones o soluciones que se presentan para los problemas que se generan en las transformaciones, pero no se ha logrado obtener un estándar que dé solución a estos problemas y que aplique en diferentes plataformas, lo que se pudo observar con la lectura de los diferentes artículos es que cada uno propone una solución y la defiende como la mejor pero ninguno tiene la última palabra, es decisión de los desarrolladores y quien desee implementar este tipo de soluciones cual se adapta de mejor manera a su entorno y su necesidad, por esto se deben analizar siempre las diferentes opciones que se pueden encontrar y algunas veces la mejor solución es la combinación de más de una de estas propuestas.

6. Referencias

Aquino, N. (2009). Adding flexibility in the model-driven engineering of user interfaces. *Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing systems - EICS '09*, 329. doi:10.1145/1570433.1570496

Bernardo, J., & Anaya, R. (2007). Marco de Referencia para la Evaluación de Herramientas Basadas en MDA, (c), 1–14.

Fincher, S., Finlay, J., Park, B., Greene, S., Jones, L., Matchen, P., … Madriqueres, P. (2003). Perspectives on HCI Patterns : Concepts and Tools.

Montero, F., Inform, S., & Mancha, L. (2010). Diseño de reglas de adaptación y transformación para interfaces de usuario Designing adaptation and transformation rules for user interface, *7*(1).

Trujillo, S., Zubizarreta, A., Mendialdua, X., & de Sosa, J. (2009). Feature-oriented refinement of models, metamodels and model transformations. *Proceedings of the First International Workshop on Feature-Oriented Software Development - FOSD '09*, 87. doi:10.1145/1629716.1629734

Vanderdonckt, J. (2008). Model-Driven Engineering of User Interfaces : Promises , Successes , Failures , and Challenges.

Vanderdonckt, J., & Simarro, F. M. (2010). Generative pattern-based design of user interfaces. *Proceedings of the 1st International Workshop on Pattern-Driven Engineering of Interactive Computing Systems - PEICS '10*, 12–19. doi:10.1145/1824749.1824753

Yu, Y., Lin, Y., Hu, Z., Hidaka, S., Kato, H., & Montrieux, L. (2012). Maintaining invariant traceability through bidirectional transformations. *2012 34th International Conference on Software Engineering (ICSE)*, 540–550. doi:10.1109/ICSE.2012.6227162