Desarrollo de Interfaces de Usuario: un método basado en modelos

User Interfaces Development: a method based on models

Juan Carlos Mejias Cruz [1]

Nemury Silega Martínez [2]

[1] **CEGEL, Centro de Gobierno Electrónico. Universidad de las Ciencias Informáticas. La Habana. Cuba**

[2] **CEGEL, Centro de Gobierno Electrónico. Universidad de las Ciencias Informáticas. La Habana, Cuba**

Resumen

El desarrollo de las interfaces de usuario es una etapa del desarrollo de software en la que se invierte un porcentaje de tiempo y recursos significativos. Actualmente esta situación se ha ido complejizando debido al aumento del número de tecnologías involucradas. Es por ello que la comunidad científica internacional se sigue esforzando en plantear soluciones orientadas a disminuir el tiempo y a aumentar la calidad de las interfaces resultantes. Entre estas propuestas se destaca el Desarrollo de Interfaces de Usuario Dirigido por Modelos (MDUID). Sin embargo, en las propuestas que siguen este paradigma se ha prestado mayor atención a las soluciones tecnológicas, dejando a un lado los aspectos metodológicos. Este aspecto ha dificultado la adopción y éxito de las propuestas basadas en MDD por parte de la industria del software. En tal sentido, el presente trabajo se plantea como objetivo desarrollar una guía para el desarrollo de interfaces de usuario dirigido por modelos que detalle los roles involucrados, las actividades a realizar por cada rol y los artefactos resultantes de cada actividad. Su aplicación contribuirá a aumentar las probabilidades de éxito en la adopción del paradigma MDD y de esta manera explotar los beneficios de este paradigma, entre los que sobresale la disminución del tiempo y la reducción de la cantidad de errores en el desarrollo de las IU durante el proceso de desarrollo de software.

Palabras clave: interfaz de usuario, MDUID, roles, actividades, modelos.

Abstract

The user interfaces development is a stage of software development in which a significant percentage of time and resources are invested. Currently, this situation has become more complex due to the increase in the number of technologies involved. That is why the international scientific community continues to strive to propose solutions aimed at reducing time and increasing the quality of the resulting interfaces. Among these proposals, the Model-Driven User Interface Development (MDUID) stands out. However, in the proposals that follow this paradigm, more attention has been given to technological solutions, leaving aside the methodological aspects. This aspect has hampered the adoption and success of proposals based on MDD by the software industry. In this sense, this paper aims to develop a guide for the model-driven user interface development that detail the roles involved, the activities to be performed by each role and the resulting artifacts of each activity. Its application will contribute to increase the probabilities of success in the adoption of the MDD paradigm and in this way exploit the benefits of this paradigm, among which stands out the decrease of time and the reduction of the number of errors in the UI development during the software development process.

Keywords: *user interface, MDUID, roles, activities, models.*

Introducción

La creación de las interfaces de usuario es un área del desarrollo de software que ha alcanzado un mayor auge a partir de la década de los setenta (Muñoz Márquez, 2007). Esta actividad tradicionalmente ha sido una labor artesanal llevada a cabo por los mismos desarrolladores de software, dependiendo de la calidad, de la habilidad y experticia. Una interfaz de usuario (IU) establece el vínculo entre el usuario y el programa de computadora que se ejecuta. Pueden adoptar diferentes formas, que van desde la simple línea de comandos hasta las interfaces gráficas que proporcionan las aplicaciones más modernas.

Crear IU bien diseñada exige dedicación pues generalmente las interfaces son grandes, complejas y difíciles de implementar, depurar y modificar. Según Rosson y Mayers (1992) un porcentaje significativo de la actividad de desarrollo de software es invertido en diseñarlas e implementarlas. Además, un estudio realizado por el Grupo Gartner revela que más del 70% del esfuerzo de desarrollo de las aplicaciones interactivas, está dedicado a la interfaz de usuario (Gartner Group, 1994).

En entrevista realizada a Jefes de Proyectos, Analistas, Arquitectos de sistema, Desarrolladores e Implantadores de soluciones de la Universidad de las Ciencias Informáticas con años de experiencia en el desarrollo de software, se obtuvo que el 100% de los entrevistados señalan que un tiempo apreciable (55% del tiempo dedicado a la Ingeniería de Requisitos) es invertido por los analistas en el diseño de prototipos no funcionales, que son utilizados únicamente para la validación de los requerimientos del sistema con el cliente. Dichos prototipos son creados nuevamente por los desarrolladores atendiendo a la plataforma concreta de desarrollo, invirtiéndose tiempo y recursos que pudieran ser aprovechados en la ejecución de otras tareas.

Por otra parte, el 34,4% de los entrevistados afirmó que han incorporado desviaciones en los cronogramas de sus proyectos por deficiencias en el desarrollo de las IU, pues el proceso no está organizado. Cuando es necesario enfrentar un cambio o actualización de tecnología en las IU, el 66,67% resalta que el

proceso puede volverse engorroso porque usualmente se toma como guía el código creado en la anterior tecnología, pero sin reutilizarse.

Frente a esta situación que se ha ido complejizando debido al aumento del número de tecnologías involucradas en el desarrollo de la interfaz de usuario, diversas aportaciones se han orientado a conseguir que su desarrollo sea en menor tiempo y con mayor calidad. Entre estas propuestas despierta gran interés el Desarrollo Dirigido por Modelos de Interfaces de Usuario (MDUID, por las siglas en inglés de Model-Driven User Interface Development) que comprende dos subáreas del desarrollo de software: el Desarrollo Dirigido por Modelos y el Desarrollo de Interfaz de Usuario.

Teniendo en cuenta las ventajas de MDD (Akiki y otros, 2015; Engel y otros, 2016; Yigitbas y otros, 2017) diversas soluciones tecnológicas se han propuesto para el MDUID. Sin embargo, los aspectos metodológicos han recibido poca atención y hoy se carecen de propuestas que faciliten la adopción por parte de la industria del software de este paradigma de desarrollo. En este sentido, el presente trabajo tiene como objetivo desarrollar una guía para el desarrollo de interfaces de usuario dirigido por modelos, que contribuya a disminuir el tiempo y a reducir la cantidad de errores en el desarrollo de las IU.

La motivación principal para realizar este trabajo consiste en ofrecer una propuesta que detalle los aspectos metodológicos (roles involucrados, actividades a realizar por cada rol y artefactos a obtener en cada actividad) que permita utilizar y explotar los beneficios del MDUID en el desarrollo de software. La estructura del documento es la siguiente: en la próxima sección se abordan los principales aspectos relacionados al MDUID por la importancia que tiene en la guía que se propone. Seguidamente, se detalla la propuesta desarrollada y se discuten los resultados con un caso de estudio. Finalmente se presentan las conclusiones del trabajo.

Materiales y métodos

MDUID. Su estado actual en Cuba.

El Desarrollo Dirigido por Modelos (MDD) es un enfoque para el desarrollo de software basado en la creación de modelos a distintos niveles de abstracción y su uso como base de un proceso de generación automática de código. Entre sus beneficios se encuentran una mayor simplicidad del proceso, mejora de la productividad del proceso de desarrollo, así como la calidad externa de las aplicaciones resultantes (Yigitbas y otros, 2017).

Dentro de MDD surge el MDUID que tiene como principal objetivo, propiciar un entorno en el que los desarrolladores puedan diseñar e implementar interfaces de usuario de una forma profesional, consistente y sistemática (Meixner, y otros, 2013; Vanderdonckt y otros, 2014; Kühn y otros, 2017). MDUID está basado en la idea de que la interfaz de usuario puede ser completamente definida por un conjunto de modelos declarativos, cada uno de los cuales trata facetas particulares de la interfaz de usuario como las tareas, la presentación, entre otros.

Los modelos guían el ciclo de vida del desarrollo de la interfaz de usuario y son la base para la generación automática de diversos productos de software, entre ellos, el código fuente de la interfaz de usuario.

Los desarrolladores trabajan sobre esos modelos realizando diversas transformaciones para finalmente obtener de forma automática (o semiautomática) modelos derivados, el código de la interfaz de usuario, documentación o diagramas.

Para conocer el estado actual del MDUID en Cuba, se realizó una búsqueda de trabajos relacionados. Se encontró un estudio realizado hasta el 2014, el cual comprobó mediante el análisis de 15 trabajos, que no existe una adecuada aplicación de los métodos más novedosos para el desarrollo de software en Cuba. Tanto la industria de software como las investigaciones relacionadas con el desarrollo de software en Cuba no se han interesado lo suficiente por el MDD, pese a que acapara gran aceptación por la comunidad de investigadores y desarrolladores de software en el mundo (Silega y otros, 2014).

El estudio demuestra también que los principales esfuerzos de los investigadores y desarrolladores de software en Cuba se centran en ofrecer soluciones para mejorar áreas como la agricultura, educación, salud y otras. No obstante, existen muy pocas propuestas para mejorar el proceso de desarrollo de software y se centran en mejorarlo por la vía de perfeccionar la organización de los proyectos (Silega y otros, 2014).

Con el objetivo de determinar la evolución de este tema desde el 2014 hasta la actualidad, se realizó una búsqueda en la Revista Cubana de Ciencias Informáticas (RCCI), por ser considerada un medio nacional importante para la difusión de investigaciones relevantes en la rama de las Ciencias Informáticas y de la Computación. La búsqueda no arrojó resultados, lo que evidencia que la adopción y aceptación del MDUID en nuestro país es aún insuficiente.

Guía para el desarrollo de IU dirigido por modelos

La guía que propone el presente trabajo describe los roles involucrados en el MDUID y delimita sus competencias y responsabilidades. Detalla las actividades a realizar por cada rol, los artefactos a obtener en cada actividad y los principios y premisas necesarios para su aplicación. Una vista general de la guía propuesta se puede apreciar en la Figura 2.

Las cuatro fases en que está dividida la guía tienen un objetivo específico y contienen un grupo de actividades a realizar y generan artefactos que constituyen entradas para fases posteriores. Las fases propuestas son las siguientes:

Fase Ingeniería Básica: El objetivo de esta fase es comprender el alcance del software y especificar las jerarquías de tareas que deben ser realizadas en/con objetos del dominio.

Fase Ingeniería de Detalle: En esta fase se realiza el diseño conceptual del software sin hacer referencia a la implementación. Se utiliza como entrada la información y artefactos generados en la fase anterior y se crean los modelos independientes de la computación.

Fase Construcción: Durante esta fase se describe de forma concreta pero independiente de la implementación cómo es percibida la interfaz por los usuarios. En esta fase se obtienen los modelos independientes de la plataforma.

Fase Implementación: Es en esta fase se obtiene la interfaz de usuario final dependiente de la implementación. Se generan los modelos dependientes de la plataforma.

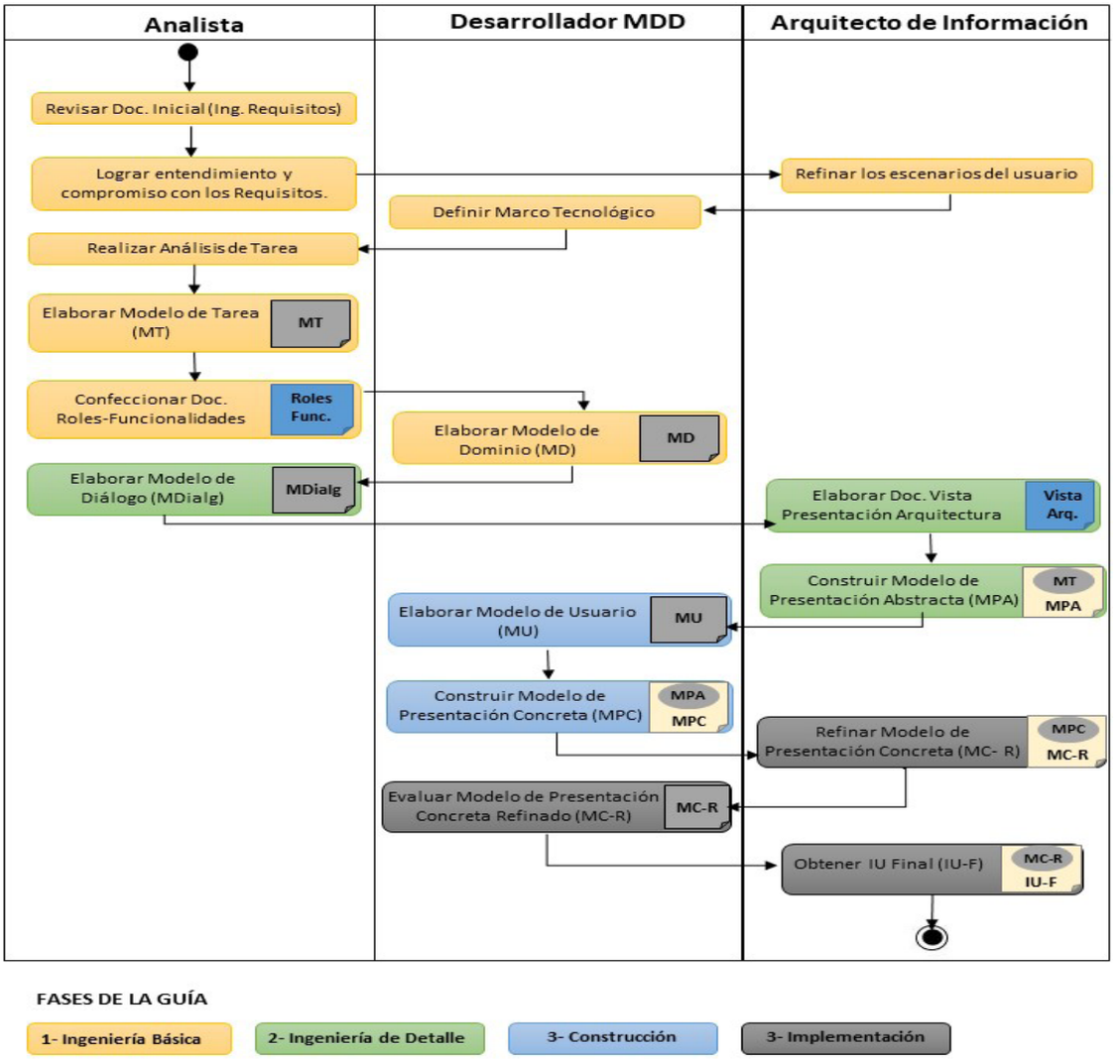

Figura 2: Flujo de actividades de la guía propuesta
Fuente: Elaboración propia

En la Figura 3 se muestra como se debe desarrollar el ciclo de vida de la guía quedando relacionadas cada una de las fases descritas.

Figura 3: Ciclo de vida de la guía propuesta

Fuente: Elaboración propia

Roles

Se definen como roles responsables para la aplicación de la guía: ***Analista, Arquitecto de información y Desarrollador MDD***. La Tabla 1 muestra un resumen de las responsabilidades de cada uno de los roles descritos anteriormente.

Tabla 1 Responsabilidades de los roles por fases de la guía

Rol	Fases de la guía	Nivel de Cameleon para el MDUID	Nivel MDD
Analista	Ingeniería Básica	Nivel Conceptos y Tareas	CIM
Arquitecto de información	Ingeniería de Detalle	Nivel IU Abstracta	PIM
Desarrollador MDD	Construcción Implementación	Nivel IU Concreta Nivel IU Final	PSM

En conjunto con los roles descritos intervienen otros roles que son transversales al proceso de desarrollo como el Jefe de proyecto y arquitectos. Dichos roles deben poseer valores humanos tales como responsabilidad y compromiso con el equipo, valores que contribuirán a crear un ambiente de trabajo positivo y a mejorar la calidad tanto del proceso de desarrollo como del producto final.

Actividades

Para definir las fases propuestas en esta guía se consideraron los objetivos específicos a cumplir en cada una de ellas. De estos objetivos se derivaron las acciones, medios y métodos necesarios para desarrollarlos, se definieron cuáles serían sus resultados esperados y los responsables de su ejecución. A continuación, se detallarán las actividades que se ejecutarán en cada una de las fases.

Fase Ingeniería Básica

Durante el desarrollo de esta fase se identificaron un conjunto de actividades para analizar el alcance establecido, los objetivos generales y la estructura organizativa, enmarcando así el proceso de desarrollo al entorno. Se definen las siguientes actividades:

Revisar de la documentación inicial: La documentación resultante de la Ingeniería de Requisitos es analizada por el equipo de proyecto, con el objetivo de detectar errores y evitar su propagación. **Responsable:** Analista con Equipo de desarrollo.

Lograr el entendimiento y el compromiso con los requisitos por parte del equipo de desarrollo: Se realizarán capacitaciones dirigidas por el analista encaminadas al entendimiento de cada funcionalidad que el software debe realizar. Para garantizar una mejor comprensión, se debe hacer uso de los artefactos ingenieriles. **Responsable:** Analista.

Refinar los escenarios del usuario: Con esta actividad se pretende refinar los escenarios del usuario, representar la forma en que los actores interactúan con el software y revisar los escenarios del usuario para verificar que estén completos. A partir de este estudio se identifican elementos de arquitectura de información, de diseño de información e incluso especificaciones de usabilidad y accesibilidad. Para esta actividad el Arquitecto de información puede emplear técnicas como Entrevista, Cuestionarios, Observación de Campo y Focus Groups, con el objetivo de llevar a cabo un proceso de indagación. **Responsable:** Arquitecto de Información.

Definir el marco tecnológico a utilizar en el desarrollo de la solución: En esta actividad se especifica el marco tecnológico a utilizar en el desarrollo de la solución en cada una de las fases, como los meta-modelos, lenguajes y herramientas tanto para el modelado como para la especificación y ejecución de las reglas de transformación. **Responsable:** Desarrollador MDD con Equipo de desarrollo.

Realizar un Análisis de Tarea: Con esta actividad se pretende identificar las acciones requeridas para completar una tarea, así como los intercambios entre los usuarios, posibilitando que la interfaz que se modele sea lo más similar posible a lo que están acostumbrados y a como lo realizan en la práctica. **Responsable:** Analista.

Elaborar Modelo de Tarea: Haciendo uso de las tecnologías especificadas se elabora este modelo. Para ello se identifican, especifican y analizan las tareas que el usuario desea llevar a cabo con el software. **Responsable:** Analista.

Confeccionar el documento Roles-Funcionalidades: Una vez se cuenta con las tareas y roles encargados de su ejecución, se elabora el documento Roles-Funcionalidades que recoge para cada rol del sistema las acciones que puede lanzar y los espacios de trabajo que puede manejar. **Responsable:** Analista

Elaborar Modelo de Dominio: Utilizando la tecnología especificada para realizar este modelo, se procede a su realización. El artefacto resultante describe los datos que los usuarios manejan empleando la interfaz de usuario. **Responsable:** Desarrollador MDD.

Fase Ingeniería de Detalle

En esta fase se definen las actividades orientadas a obtener el diseño conceptual del software sin hacer referencia a la implementación. Se especifican las siguientes actividades:

Elaborar Modelo de Diálogo: Cumpliendo con las tecnologías aprobadas en el marco tecnológico, se elabora el Modelo de Diálogo que describe las posibles conversaciones entre la IU y el usuario. **Responsable:** Analista.

Confecciona el documento Vista de Presentación de la Arquitectura que define los elementos de apariencia y usabilidad de las interfaces de usuario. **Responsable:** Arquitecto de Información.

Construir el Modelo de Presentación Abstracta: Haciendo uso de la notación y herramienta especificada, se construye el Modelo de Presentación Abstracta, el cual contiene los componentes de la interfaz de usuario, su disposición y su apariencia. Explotando los beneficios que brinda el MDUID, este modelo puede ser generado automáticamente a partir del Modelo de Tarea, validando el artefacto de entrada.

Responsable: Arquitecto de Información.

Fase Construcción

En esta fase se detallan las actividades para obtener de forma concreta pero independiente de la implementación la interfaz de usuario. Las actividades a realizar son:

Elaborar Modelo de Usuario: Empleando las tecnologías seleccionadas, se elabora el Modelo de Usuario, donde los usuarios se organizan en grupos que tienen roles comunes para la aplicación. Son componentes de este modelo grupos, usuarios y sus eventos asociados. Resulta relevante para la construcción de este modelo, el artefacto Documento Roles-Funcionalidades. **Responsable:** Desarrollador MDD.

Construir el Modelo de Presentación Concreta: Haciendo uso de las tecnologías especificadas, se construye el Modelo de Presentación Concreta. Explotando los beneficios del MDUID en cuanto a la automatización de actividades, este modelo puede ser generado a partir del Modelo de Presentación Abstracta. Para ello se debe validar dicho modelo manualmente o haciendo uso de alguna herramienta existente. Esta separación en nivel abstracto y concreto del modelo de presentación permite una generación de la interfaz de usuario para distintas plataformas a partir de una misma descripción abstracta de la interfaz. **Responsable:** Desarrollador MDD

Fase Implementación

En esta fase como principal resultado se obtiene la interfaz de usuario final. Las actividades a realizar son las siguientes:

Refinar el Modelo de Presentación Concreta: Haciendo uso de las tecnologías especificadas, se refina el Modelo de Presentación Concreta obtenido. Esta actividad garantiza validar los componentes en relación al modo de interacción, así como su distribución en la interfaz de usuario y el cumplimiento con los elementos definidos en el artefacto Vista de Presentación de la Arquitectura. **Responsable:** Arquitecto de Información.

Evaluar el Modelo de Presentación Concreta refinado con usuarios reales: Con esta actividad se persigue evaluar que el Modelo de Presentación Concreta refinado esté completo, consistente y sin omisiones. El arquitecto de información puede utilizar diferentes técnicas del DUX. Como salida de esta actividad se tiene el modelo refinado y enriquecido con la opinión de los usuarios reales. **Responsable:** Arquitecto de Información.

Obtener la IU Final: Atendiendo a las tecnologías especificadas en el marco tecnológico, se obtiene a partir del Modelo de Presentación de IU Concreta refinado, la IU Final para una plataforma específica de desarrollo. Para ello se debe validar dicho modelo manualmente o haciendo uso de alguna herramienta existente. **Responsable:** Desarrollador MDD.

Artefactos

Los artefactos que se proponen y son resultados de las actividades a ejecutar en cada una de las fases propuestas son:

- Modelo de Tarea
- Documento Roles-Funcionalidades
- Modelo de Dominio
- Modelo de Presentación Abstracta
- Modelo de Diálogo
- Vista de Presentación de la Arquitectura
- Modelo de Usuario
- Modelo de Presentación Concreta

Resultados y discusión

Como caso de estudio para demostrar la aplicabilidad de la guía propuesta, se seleccionó el Sistema de Gestión para la Atención a la Población (SIGAP), producto de software en desarrollo por el Centro de Gobierno Electrónico (CEGEL) de la Facultad 3. Se tuvieron como artefactos de entrada:

1. CEGEL_SIGAP_Especificacion_de_requisitos_funcionales
2. CEGEL_SIGAP_Especificacion_de_requisitos_no_funcionales
3. CEGEL_SIGAP_Historias_de_usuario

Una vez se tienen estos documentos aprobados por el cliente se procedió a aplicar la guía propuesta.

Como resultado de la fase Ingeniería Básica se revisó cada artefacto por el equipo conformado para la aplicación de la guía. Aunque no se encontraron errores que pudieran propagarse, si se realizaron modificaciones en las descripciones de algunas funcionalidades para una mejor comprensión. Además, se efectuaron tres encuentros de capacitaciones, donde se explicó el objetivo del sistema basado en la descripción del negocio, se analizaron cada uno de los requisitos especificados y se realizó un debate sobre las posibles tecnologías a utilizar. Por otra parte, se aplicó una entrevista a los especialistas del Área de Atención a la Población de la ANPP para refinar cada uno de los escenarios del usuario que se tenían especificados.

En la definición del marco tecnológico a utilizar para el desarrollo de la solución fue de vital importancia los debates generados en los encuentros de capacitaciones realizados. Como resultado de estas actividades y teniendo en cuenta el alcance definido del proyecto a desarrollar, se seleccionaron las siguientes tecnologías:

1. Plataforma Eclipse Modeling Framework.
2. Se construirán los meta-modelos propios de cada uno de los modelos que se proponen.

3. ATL se utilizará como lenguaje de transformación.

Posteriormente se realizó un análisis de las tareas identificadas para garantizar que las IU sean lo más similar posible a como se realizan las actividades en la práctica cotidiana por los usuarios. Una vez el ambiente de desarrollo quedó instalado, se comenzaron a ejecutar las actividades propias del proceso de desarrollo, generando los artefactos especificados en la guía. Para elaborar el modelo de Tarea fue necesario elaborar el metamodelo de tarea (MMTarea.ecore) para así cumplir los principios básicos del paradigma de desarrollo en el cuál se sustenta la guía que se propone. Basado en este metamodelo, se conformó el Modelo de Tarea del SIGAP. Ambos artefactos se muestran en la Figura 4.

Para una mejor comprensión del Modelo de Tarea elaborado, se decidió representar las tareas de mayor relevancia haciendo uso de la notación ConcurTastTrees (CTT) en la herramienta ConcurTastTrees Environment (CTTE). Dicha especificación se muestra en la Figura 5.

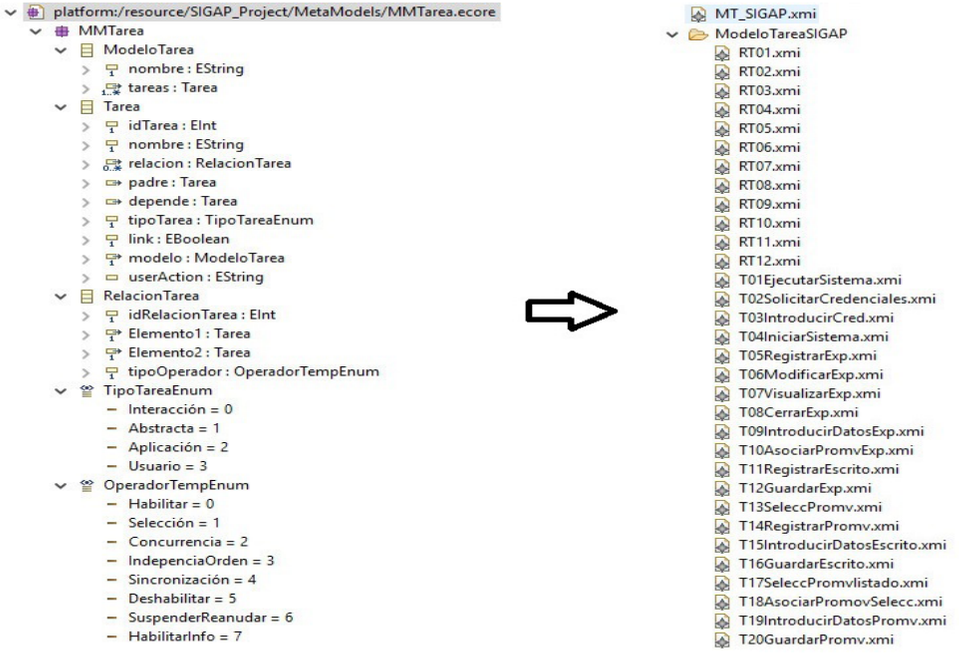

Figura 4: Metamodelo y modelo de Tarea

Fuente: Elaboración propia

Una vez elaborado y validado el modelo de tarea, se confeccionó el documento Roles-Funcionalidades, que recoge para cada rol las acciones que puede lanzar y los espacios de trabajo que puede manejar. Seguidamente se elaboró el modelo de Dominio para el cual fue necesario elaborar el metamodelo de dominio (MMDominio.ecore). Basado en este metamodelo, se conformó el Modelo de Dominio del SIGAP.

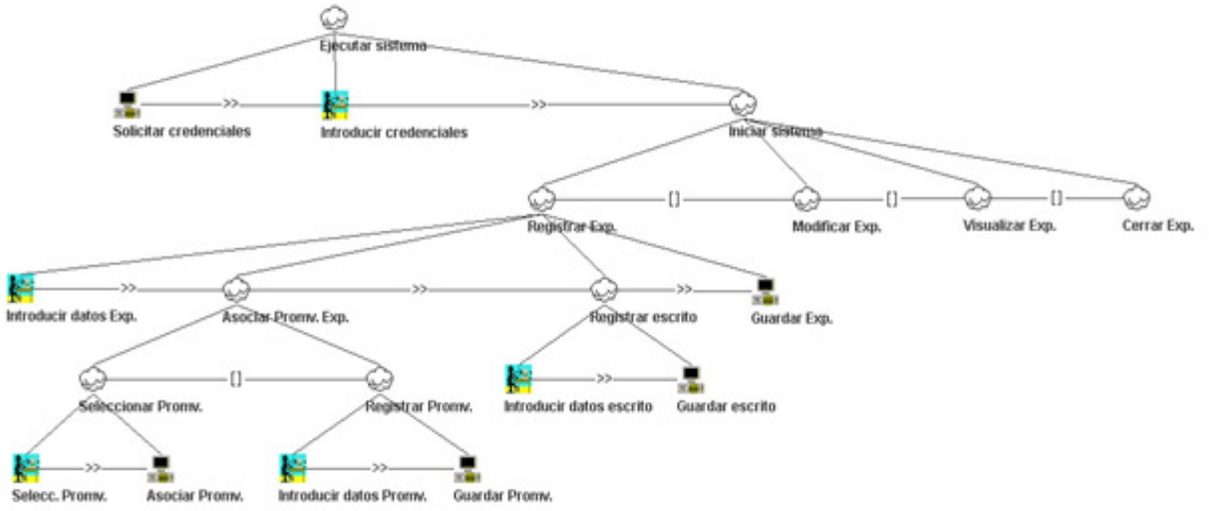

Figura 5: Especificación de un fragmento del Modelo de Tarea con CTT
Fuente: Elaboración propia

Se elaboró también el Modelo de Diálogo siendo necesario construir el metamodelo de diálogo para luego obtener el Modelo de Diálogo del SIGAP y se confeccionó el artefacto Vista de Presentación de la Arquitectura que define los elementos de apariencia y usabilidad de las interfaces de usuario.

Para aprovechar los beneficios del MDUID y obtener modelos de forma automática se definieron e implementaron reglas de transformación en ATL. La Figura 6 muestra dos de las reglas de transformación ejecutadas para obtener el Modelo de Presentación Abstracta del Modelo de Tarea presentado. Para ellos fue necesario transformar las tareas en especificadas en el Modelo de Tarea en Objetos de Interacción Abstractos (AIO) que son los componentes fundamentales del Modelo de Presentación Abstracta. En la Figura 7 se muestra el modelo resultante de la ejecución de las dichas reglas basado en el metamodelo creado.

```
helper def: task: MMTarea!Tarea =   OclAny;

helper context MMTarea!Tarea def: isTaskAbstract(): Boolean =

helper context MMTarea!Tarea def: isTaskInteraction(): Boolean =

helper context MMTarea!Tarea def: isTaskApplication(): Boolean =

helper context MMTarea!Tarea def: isTaskUsser(): Boolean =


entrypoint rule CreatingModel() {
    to
        cont: MMPresentAbst!ModeloPresentAbst (
            nombre <- 'Modelo Presentacion Abstracto SIGAP'
        )
    do{
        thisModule.modeloPA <- cont;
    }
}

rule Task2AIO {
    from
        t: MMTarea!ModeloTarea (
            t.oclIsTypeOf(MMTarea!ModeloTarea)
        )
    to
        pa: MMPresentAbst!ModeloPresentAbst (
            aios <- about
        ),
        about: distinct MMPresentAbst!AIO foreach(recor in t.tareas) (
            idAIO <- recor.idTarea,
            nombre <- recor.nombre
        )
```

Figura 6: Reglas de transformación especificadas en ATL

Fuente: Elaboración propia

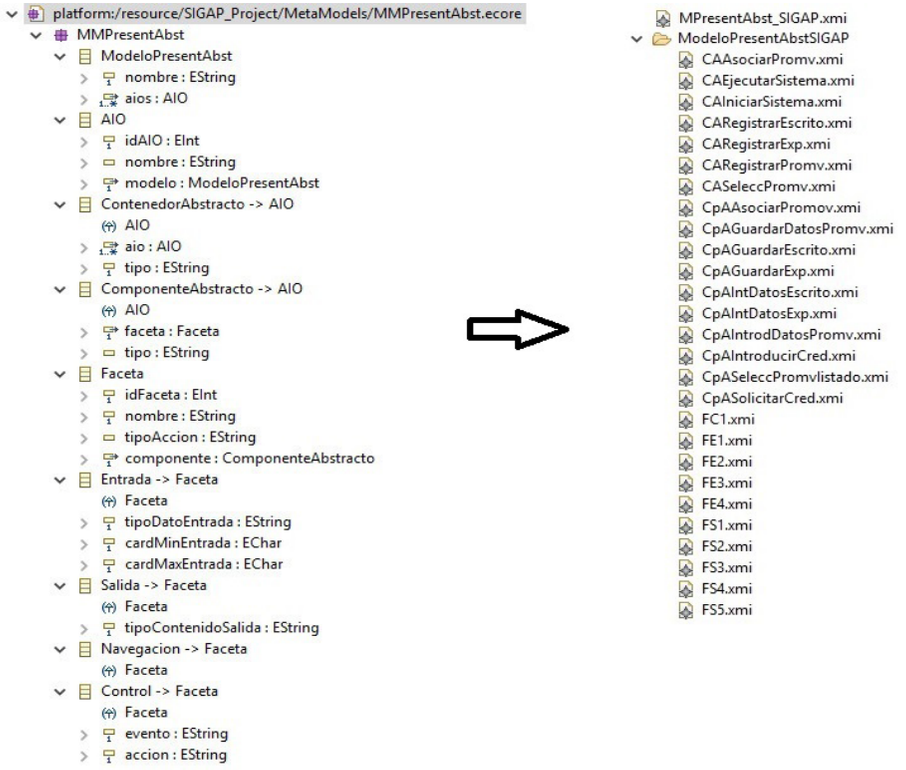

Figura 7: Metamodelo y modelo de Presentación Abstracta
Fuente: Elaboración propia

Actualmente se están implementando las reglas de transformación para generar automáticamente a partir del Modelo de Presentación Abstracta el Modelo de Presentación Concreta. Posteriormente se debe refinar este modelo para generar la IU Final en tecnologías para la web.

Como principales resultados alcanzados hasta el momento se tiene un aumento de la productividad, la usabilidad y la mantenibilidad. En cuanto al aumento de la productividad puede mencionarse que, si bien se invierte tiempo en el proceso de modelado inicial, luego el tiempo se gana cuando se obtienen de formas automáticas los modelos de presentación abstracta, concreta y la IU Final.

Se aprecia una mejora considerable en la usabilidad resultante pues al aplicar técnicas tanto del diseño centrado en el usuario como del diseño de la experiencia de usuario, se crea un estado emocional placentero en la interacción con el sistema. Por ejemplo, refinar los escenarios del usuario y luego realizar un análisis de tareas posibilitó que la interfaz que se modele sea lo más similar posible a como se realizan en la práctica las tareas que se informatizaron. Con respecto a la mantenibilidad, se obtienen resultados alentadores pues las interfaces que se obtienen son menos complejas y más modulares. Además, el proceso de mantenimiento se simplifica, ya que al existir artefactos que se generan automáticamente, cuando es necesario realizar un cambio, por ejemplo, en el Modelo de tareas, se generan nuevamente los modelos que de él se derivan, disminuyendo así el impacto del cambio y la cantidad de errores.

Conclusiones

La revisión de los aspectos asociados al MDUID permitió corroborar la importancia de este paradigma en empresas desarrolladoras de software en cuanto al aumento de productividad, reutilización y calidad. Mediante una búsqueda bibliográfica se pudo concluir que la adopción y aceptación del MDUID en nuestro país es aún insuficiente.

A partir de los elementos teóricos estudiados se elaboró una guía para el desarrollo de interfaces de usuario dirigido por modelos. Durante su definición se tuvieron actividades y técnicas del DCU, AI, DUX y los aspectos recogidos en las entrevistas realizadas, lo que demuestra su nivel de aplicabilidad. Como los elementos más importantes que la guía incluye se encuentran el flujo de actividades de las cuatro fases definidas, los roles responsables de realizar cada actividad, así como los artefactos resultantes.

Para demostrar su nivel de aplicabilidad se describieron los resultados alcanzados hasta el momento con un caso de estudio lo que demuestra mejoras en la productividad, usabilidad y mantenibilidad del sistema desarrollado. La guía que se propone puede ser el paso inicial para el desarrollo de una herramienta posterior que soporte todo el ciclo de vida del desarrollo de un proyecto.

Referencias

Akiki, P. A., A. K. Bandara and Y. Yu (2015). Adaptive model-driven user interface development systems. ACM Computing Surveys (CSUR).

Engel, J., C. Herdin and C. Martin (2016). Evaluation of model based user interface development approaches. Springer, Cham: 295–307.

Gartner Group. 1994. Informe Gartner Group: Simposium anual del futuro de la tecnología de los sistemas de información. Cannes: s.n., 1994.

Inc., OCI (2004). Object Computing Inc. (OCI) “Where the object is your success!”

Kühn, M. and P. Forbrig (2017). Adapting user interface models by transformations based on UI patterns. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). 10271: 456-466.

Meixner, Gerrit, Calvary, Gaelle and Vanderdonckt, J. 2013. Introduction to model-based user interfaces. Technical report, MBUI working group. 2013.

Muñoz Márquez, Francisco Javier. 2007. ACAUI: Abstracción de interfaces de usuario a partir de especificaciones concretas. Castilla: s.n., 2007.

Rodríguez Sánchez, T. (2015). Metodología de desarrollo para la Actividad productiva de la UCI. Universidad de las Ciencias Informáticas, La Habana, Cuba.

Rosson, Mary Beth and Myers, Brad A. 1992. Survey on user interface programming. Computer/Human Interac-

tion. 1992.

Silega Martínez, N. (2014). Método para la Transformación automatizada de modelos de procesos de negocio a modelos de componentes para Sistemas de Gestión Empresarial. La Habana, Cuba. Ediciones Futuro.

Silega, Nemury, y otros. 2014. Estado de la complejidad arbitraria y Arquitectura Dirigida por Modelos en el desarrollo de software en Cuba. La Habana: Ediciones Futuro, 2014. ISSN: 2227-1899.

Vanderdonckt, J., R. Tesoriero, y otros. (2014). MBUI - abstract user interface models. Technical report, World Wide Web Consortium (W3C).

Yigitbas, E., H. Stahl, S. Sauer y G. Engels (2017). Self-adaptive UIs: Integrated model-driven development of UIs and their adaptations. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). 10376 LNCS: 126-141.