REVECOM

REVISTA VENEZOLANA DE COMPUTACIÓN

ML • Java • Python • C# • C++ • Scala
Perl • Haskell • Lisp • JavaScript

Sociedad Venezolana de Computación

Vol. 4, No. 1
Junio 2017

ISSN: 2244-7040

SVC VENEZUELA
Sociedad Venezolana de Computación

Revista
Venezolana
de Computación

REVECOM

Revista Venezolana de Computación

Sociedad Venezolana de Computación

Editores:
Eric Gamess, Wilmer Pereira, Yudith Cardinale

ISSN: 2244-7040

Vol. 4, No. 1
Junio 2017

Editorial

Informática Médica

La informática médica cubre los aspectos básicos de las técnicas informáticas aplicadas a la adquisición, almacenamiento, recuperación, tratamiento y uso de la información médica. Las herramientas informáticas para la salud incluyen la gestión de toda la información relacionada con el paciente y el flujo de información que se deriva de su diagnóstico, tratamiento y prognosis; información relacionada con la práctica médica, guías de práctica clínica, terminología médica formal y disponibilidad en diferentes medios.

La informática médica tiene un ámbito de aplicación amplio que abarca la enfermería, gestión hospitalaria, atención clínica, odontología, farmacia, salud pública, terapia ocupacional e investigación biomédica. Desde hace poco, se aplica en otros campos relacionados con la salud, como compañías de seguros, proveedores, organizaciones de servicios y de consultoría. El personal requerido en esta disciplina reúne la asistencia de salud, el personal de informática y un nuevo cargo profesional conocido como gestores de información de salud. Estos últimos se encargan de los sistemas de información de pacientes y aseguran que la información sea adecuada, accesible, confidencial y segura.

La informática médica comenzó en 1960 e inicialmente no recibió buena aceptación por parte de la administración, médicos y otros profesionales de salud. Luego se produjo una rectificación de esta visión, cuando tomó un carácter menos instrumental y más relacionado con el contexto de la información. Así se involucra a los médicos con la contribución de esta nueva disciplina a sus prácticas y se motiva a la administración de los hospitales a implicarse con estas nuevas prácticas. Actualmente, la informática médica ha surgido y ya su rol no se discute. La realidad actual refleja:

- El incremento del conocimiento médico y los parámetros necesarios para atender a los pacientes.
- El surgimiento de nuevas técnicas en imagenología médica, exploraciones funcionales o técnicas derivadas de la ingeniería biológica que contribuyen al aumento de conocimiento.
- Las dificultades de los médicos para gestionar todo este conocimiento y adaptarlo a la atención médica en un marco de tiempo adecuado.
- Las historias clínicas no son una mera colección de notas, ya que incluyen además imágenes (ecografía, angiografía digital, resonancia magnética), señales (eléctricas, acústicas, electrofisiológicas), conceptos interconectados (conocimientos fisiopatológicos) y protocolos.
- Las enfermedades crónicas son cada vez más comunes en la práctica médica y requieren una atención creciente además de involucrar más especialistas en su cuidado.
- El cuidado del paciente no depende de un único médico o un pequeño número de médicos, sino de un equipo de cuidado que comparte recursos y habilidades complementarias.
- Los registros de pacientes deben ser fácilmente accesibles y comunicables a diferentes miembros del equipo de atención.
- La existencia de restricciones económicas, legales y éticas.

Entre algunas de las áreas que apoyan la informática médica existen tres grandes grupos: (1) Los sistemas de información sanitaria (sistemas de información médica, bancos de datos computarizados, automatización de laboratorios clínicos, radiología y cuidados intensivos); (2) Apoyo a la decisión y aseguramiento de la calidad (la toma de decisiones asistida por computador, los sistemas de gestión centrados en el paciente, el aseguramiento y mejora de la calidad); (3) Registros médicos basados en computadores y sistemas integrados de información (estándares y normas de información médica, investigación y desarrollo, coordinación, privacidad de la información de salud).

Editorial

La investigación en informática médica está comprometida con el desarrollo de métodos y herramientas para facilitar el tratamiento racional, rápido y confiable de la información médica. Cuatro palabras claves guían esta investigación: (1) *memorización,* (2) *comunicación,* (3) *soporte a la decisión* y (4) *evaluación*.

La *memorización* concierne particularmente los datos del paciente almacenados en registros médicos y el conocimiento médico de naturaleza práctica como protocolos de atención de problemas médicos individualizados, o más general como los conocimientos en anatomía, fisiología o farmacología. También se refiere al conocimiento del entorno de trabajo, tales como los recursos humanos o materiales, o el marco legal o jurídico del cuidado de la salud. La disponibilidad, fiabilidad y pertinencia de los datos que tienen que ver con un paciente o el conocimiento necesario para la toma de decisiones afectan la calidad del cuidado.

La actividad médica requiere un esfuerzo de *comunicación*. El almacenamiento de la información necesaria para la toma de decisiones sólo tiene sentido si la información almacenada es inmediatamente accesible por todo el equipo de cuidado. Sin embargo, la información sobre un determinado paciente generalmente se encuentra dispersa en una o más instituciones de salud y es importante proporcionar a los diferentes usuarios una visión única de un registro médico distribuido físicamente.

El *soporte a la decisión* asistida por computador ha sido considerado por la comunidad médica como relevante, pero de estricto campo de investigación. Sin embargo, los métodos simples de apoyo a la decisión, tales como alarmas activadas de forma automática mediante la actualización de los registros médicos han mostrado rápidamente su utilidad. Los sistemas basados en conocimiento se han orientado mejor y se han complementado con técnicas convencionales basadas en exploración estadística o probabilística de bases de datos de registros médicos y con técnicas modernas basadas en minería de datos para la extracción de conocimiento útil y su integración a los sistemas de apoyo al diagnóstico.

Las restricciones económicas así como también la ética profesional requieren de la implementación de procedimientos de *evaluación,* por ejemplo el análisis del contenido informativo o decisorio de los datos médicos, la evaluación de los procedimientos diagnósticos y terapéuticos utilizados de forma rutinaria o la evaluación de nuevas tecnologías.

La informática médica tiene unos cuantos desafíos por delante, en los cuales se encuentran:

- La mejora de la estandarización del vocabulario clínico y su codificación.
- La mejora de la integración, la comparación y la confidencialidad de los datos del cuidado de la salud a través de los sistemas informáticos y centros de salud.
- La vinculación de los datos del cuidado del paciente y la información médica adecuada y precisa para apoyar la toma de decisiones clínicas, mientras se protege la confidencialidad del paciente.
- La identificación y superación de las barreras para el uso de herramientas de informática médica en la comunidad médica y el uso productivo de la integración de información y los sistemas informáticos y de comunicación en la atención de salud.

Dra. Ana Isabel Aguilera
Profesora titular del Departamento de Computación, FACYT
Universidad de Carabobo

Revista Venezolana de Computación

ReVeCom (Revista Venezolana de Computación) es la primera revista venezolana arbitrada, periódica, digital, orienta a la publicación de resultados de investigación en el campo de la computación. ReVeCom fue creada por la SVC (Sociedad Venezolana de Computación) y tiene entre sus objetivos hacer conocer los trabajos de alta calidad investigativa que se realizan a nivel nacional, latinoamericano e internacional. La revista permite la divulgación de artículos con aporte original en castellano o inglés.

ReVeCom es una revista abierta para una mayor difusión de los resultados de investigación. Cuenta con una página web (http://www.svc.net.ve/revecom), donde se encuentran los trabajos publicados e información sobre la revista. La revista promueve la pluralidad de intereses, dando cabida a la divulgación de trabajos de todos los campos del conocimiento inherentes a la computación.

Este volumen de ReVeCom (Vol. 4, No. 1) corresponde a artículos de investigación en el campo de la computación que fueron seleccionados a través de un riguroso arbitraje por expertos del área. En esta oportunidad, el comité evaluador fue compuesto por:

Directorio de la Sociedad Venezolana de Computación

Presidente:
Dr. Leonid Tineo (Universidad Simón Bolívar)

Vicepresidente:
Dra. Yudith Cardinale (Universidad Simón Bolívar)

Secretario:
Dr. Wilmer Pereira (Universidad Católica Andrés Bello)

Tesorero:
MSc. Rosseline Rodríguez (Universidad Simón Bolívar)

Coordinadora de Educación e Investigación:
Dra. Dinarle Ortega (Universidad de Carabobo)

Coordinador de Publicaciones:
Dr. Eric Gamess (Universidad Central de Venezuela)

Coordinadora de Eventos:
MSc. Nataly Carmona (Universidad Marítima del Caribe)

Edición

Comité Editorial

Miembros del Comité Editorial

Tabla de Contenido

Revista Venezolana de Computación
ISSN: 2244-7040
http://www.svc.net.ve/revecom
Vol. 4, No. 1, pp. 1-9, Junio 2017
Fecha recepción: 27/04/2017, Fecha aceptación: 05/08/2017

Implementación de un Método para la Clasificación Automática de Documentos Usando Tareas de Procesamiento de Lenguaje Natural y un Algoritmo de Máxima Entropía

Luis Molina[1], Javier Maldonado[1]
lm.molinab@gmail.com, jmaldo@unet.edu.ve

[1] Lab. de Computación de Alto Rendimiento (LCAR), Universidad Nacional Experimental del Táchira, Venezuela

Resumen: La presente investigación tiene como propósito la implementación de un método para la clasificación automática de documentos usando una aproximación de procesamiento de lenguaje natural (PLN), y un algoritmo basado en el principio de máxima entropía. Se consiguió una combinación de técnicas y parámetros que brinde la mayor eficacia posible en la clasificación de un texto ingresado con base en un conjunto de categorías preseleccionadas. El proceso investigativo inicia con la selección de las categorías y datos a usarse para los experimentos, obteniendo las cantidades de documentos que brinden mayor estabilidad al sistema seguido del pre-procesado de dichos datos mediante el uso de algoritmos de PLN, posteriormente se ejecuta el entrenamiento y luego las pruebas a cada experimento con las cuales se obtienen las medidas de evaluación para el clasificador. Finalmente se realiza un análisis comparativo de los resultados, determinando así la combinación de parámetros y técnicas de pre-procesado que brinde mayor eficacia en la clasificación para el conjunto de documentos estudiados. Todo este proceso está enmarcado en un entorno de noticias digitales, en el cual se consiguió una clasificación efectiva para el 91% de los documentos analizados, utilizando siete categorías con un total de 1400 noticias de entrenamiento por cada una y haciendo uso de la eliminación de palabras vacías y el *stemming* como técnicas de PLN, mostrando así la efectividad de los métodos utilizados en cuerpos de texto escritos en el idioma español.

Palabras Clave: Clasificación Automática de Documentos; Procesamiento de Lenguaje Natural; Máxima Entropía.

Abstract: The main purpose of this research is to obtain a method for automatic document classification using natural language processing (NLP), and an algorithm based on maximum entropy principle. The main goal is to obtain a combination of techniques and parameters that provide the greatest possible efficiency in text classification on a set of pre-selected categories. The research begins with the selection of the categories and data to be used for the experiments, obtaining the quantities of documents that provide greater stability to the system, followed by the data preprocessing through the use of NLP algorithms, then the training is performed based on a series of parameters, after that, some tests are executed to each experiment and we obtain the evaluation measures for the classifier. Finally, a comparative analysis of the results is carried out, in that way determining the combination of parameters and preprocessing techniques that provide the highest efficiency in the classification for the studied dataset. This entire process is focused on a digital news source, which achieve a 91% classification effectiveness of the processed documents using seven categories, with a training dataset of 1400 digital news for each one and using stop word removal and stemming as NLP techniques, showing the effectiveness of this methods on text bodies written on Spanish.

Keywords: Automatic Document Classification; Natural Language Processing; Maximum Entropy.

I. INTRODUCCIÓN

Con el constante aumento de la información en formato digital, surge la necesidad de encontrar métodos que permitan obtener conocimiento útil a partir de este gran número de datos. Una de las áreas encargadas de este tipo de análisis es la minería de texto, la cual consiste en realizar operaciones para analizar textos con la finalidad de extraer conocimiento como se cita en [1]; dentro de ella se encuentra la clasificación automática de documentos, la cual ha presentado un activo campo investigativo en los últimos años, en ésta se busca tomar un gran conjunto de textos y asignarlos a una o varias categorías.

Es necesario conocer que el proceso base para la clasificación automática de documentos consta de dos fases, en la primera, el usuario define las categorías en las cuales está interesado y proporciona al clasificador un conjunto de documentos de entrenamiento, con la finalidad de que el algoritmo aprenda. Después de esto, se procede a la fase de pruebas, en la cual se le entrega al clasificador un conjunto de documentos distinto al de entrenamiento, con el objetivo de que sean categorizados y de esta forma verificar que tan precisa es la clasificación.

Este proceso de clasificación depende directamente del contenido del documento que se busca categorizar, por lo tanto, es común agregar una fase previa de pre-procesamiento de los documentos, con la intención de ajustarlos de una mejor manera al proceso de clasificación que será utilizado en las dos fases posteriores (entrenamiento y pruebas), consiguiendo finalmente mejorar los resultados del clasificador. Esto se puede lograr aplicando distintas técnicas de procesamiento de texto de acuerdo a la necesidad del analista.

En la presente investigación se hace uso de un clasificador basado en la teoría de máxima entropía, así como la utilización de una serie de métodos de procesamiento de lenguaje natural (PLN), aplicados al texto como son el *stemming* y la eliminación de palabras vacías.

Uniendo el enfoque de uniformidad de la máxima entropía con el procesamiento de los documentos previo a su clasificación, se busca aumentar el rendimiento y mejorar los resultados arrojados por el algoritmo clasificador, indicando en última instancia la combinación de métodos de PLN que brinde los mejores resultados en el proceso de categorización para el conjunto de datos empleado.

Para el cuerpo de documentos estudiado, con un total de 1400 noticias de entrenamiento por cada una de las siete categorías seleccionadas, se obtuvo un 91% de efectividad en la clasificación, seleccionando la eliminación de palabras vacías y el *stemming* como técnicas de PLN para el pre-procesado de los documentos. De esta manera se consigue mostrar la efectividad de los métodos y técnicas utilizadas para conjuntos de documentos escritos en el idioma español, ya que la mayor parte de los trabajos previos en esta área han sido realizados sobre cuerpos de texto en inglés.

El artículo tiene la siguiente organización: en la Sección II se presentan algunas definiciones importantes para la comprensión de la investigación, en la Sección III se hace referencia a trabajos previos relacionados con esta investigación, en la Sección IV se indica la metodología empleada, en la Sección V se describen los experimentos realizados y el procedimiento utilizado para conseguir los resultados a ser analizados en la Sección VI, finalmente se presentan las conclusiones respectivas y algunos apuntes para trabajos futuros.

II. Conceptos Importantes

A. Procesamiento de Lenguaje Natural (PLN)

El PLN [2] es un conjunto de métodos de inteligencia artificial enfocados al entendimiento de la lingüística, su principal objetivo consiste en procesar un conjunto de frases en lenguaje humano (también llamados lenguajes naturales), de tal forma que puedan ser interpretados automática y eficientemente por un algoritmo computacional, dando respuestas con base en las instrucciones recibidas.

De acuerdo con [3], el uso de estos lenguajes naturales, facilita el desarrollo de programas que realicen tareas relacionadas con el lenguaje o bien, desarrollar modelos que ayuden a comprender los mecanismos humanos relacionados con el lenguaje.

El PLN posee una variada gama de aplicaciones, entre estas encontramos el análisis de sentimientos, traducción automática, recuperación de información y el resumen automático de textos.

Dentro de la gran variedad de tareas de PLN existentes, algunas de las más habituales y de las cuales se hace uso en la presente investigación son: reconocimiento de nombres [4], *stemming* [5] y eliminación de palabras vacías [6].

B. Clasificación Automática de Documentos

En un entorno comprendido por datos no estructurados como son los textos planos, la clasificación o categorización automática de documentos es considerada en [7] como probablemente el tema más común al analizar datos complejos.

Partiendo de un conjunto de categorías preestablecido, el clasificador asigna una de ellas al texto analizado con base en su contenido. Tal y como se indica en [8], el principal enfoque para el problema de categorización de textos se basa en técnicas de aprendizaje automático.

Este proceso está dividido en tres etapas las cuales son:

1) Pre-procesado: como se describe en [9], en esta etapa se obtienen las características o términos clave a partir de documentos almacenados y mejorar la relevancia entre palabra y documento, así como también entre palabra y categoría. En esta etapa, el texto es procesado y se determinan aquellas entidades que proporcionen información relevante al clasificador.

2) Entrenamiento: el usuario provee al clasificador un conjunto de textos previamente etiquetados, cada uno puede ser asociado a una o más categorías. A partir de esta información, el algoritmo de aprendizaje podrá realizar asociaciones, las cuales determinarán a que categoría pertenece un documento. En [10] se describe que la categorización solo cuenta las palabras que aparecen y, a partir de las cuentas, identifica los temas principales que cubre el documento.

3) Clasificación: De acuerdo con [11], aquí se introducen nuevos documentos no revisados por el clasificador, el cual retorna una asociación de clase a la que pertenece cada uno de ellos en función de las reglas previamente generadas.

C. Medidas de Evaluación en Algoritmos de Clasificación

Ante la diversidad de algoritmos que permiten la categorización automática de una serie de documentos, es necesario que existan medidas que permitan evaluar la exactitud de las operaciones de clasificación. Los más comunes son:

1) Precisión: Es una medida que evalúa la probabilidad de que las clasificación de un documento d en una clase c sea

correcta. También se describe en [12] como la fracción de los documentos recuperados en la clasificación que son relevantes.

2) Cobertura: También conocida por su nombre en inglés recall, es una medida que evalúa la proporción de documentos d que forman parte de una clase c, que son seleccionados por el algoritmo como pertenecientes a dicha clase. Se puede definir de igual manera como la fracción de los documentos relevantes que se recuperan en la clasificación, tal como lo hace [12].

3) Medida F: Es una medida que une la precisión y la cobertura para determinar la eficiencia del algoritmo, además, tal como indica [13], incluye el parámetro β ($0 \leq \beta \leq \infty$) para indicar el nivel de importancia relativo entre ambas medidas; si β es mayor a 1 se le da un mayor peso a la cobertura, mientras que si es menor a 1, la precisión será la que tenga más relevancia y si el valor de β es 1, ambas tendrá las misma importancia. La fórmula para calcular la medida F se representa en la Ecuación (1).

$$F_\beta = \frac{(\beta^2 + 1) precision * cobertura}{\beta^2 precision + cobertura} \quad (1)$$

Para los experimentos realizados en este trabajo el valor de β utilizado es 1, los motivos de tal selección están detallados en la fase experimental.

D. Máxima Entropía

Maxent, o máxima entropía, es una técnica para determinar distribuciones de probabilidad a partir de un conjunto de datos. Su principio base establece que ante el desconocimiento de información, la distribución deberá ser uniforme.

Con relación a esto, y según se define en [14], la idea detrás de la máxima entropía es que se debe preferir los modelos más uniformes que al mismo tiempo satisfagan cualquier restricción dada. Además, en su formulación más general, la máxima entropía puede ser usada para estimar cualquier distribución de probabilidad.

En un sistema que se encuentra sujeto a una serie de restricciones, la distribución se adaptará a las mismas, manteniendo siempre la mayor uniformidad posible en la distribución de probabilidad calculada, esto es, tener máxima entropía.

Para emplear la máxima entropía en la categorización automática de documentos, se debe usar un conjunto de textos previamente etiquetados, a partir de las cuales, el algoritmo creará las restricciones necesarias para estimar la distribución de probabilidad del modelo. Un ejemplo ilustrativo acerca del uso de la máxima entropía en el ámbito de la clasificación automática de documentos se puede observar en [14].

III. TRABAJOS RELACIONADOS

Con base en la revisión de literatura relevante a la presente investigación, se ha evidenciado que existen distintos trabajos realizados previamente referentes a la clasificación automática de documentos, procesamiento de lenguaje natural y máxima entropía, aplicadas a la obtención de información.

En [15] se presenta un enfoque diferente al tradicional para construir sistemas de extracción de información. La característica principal de la arquitectura es el escaso uso de recursos lingüísticos, los cuales son reemplazados por métodos de aprendizaje supervisado. Además se presentan algunas bases acerca de los métodos usados para la clasificación automática de documentos así como las distintas medidas para evaluarlos.

En [6] se realiza un estudio acerca de la aplicación de distintas técnicas de procesamiento de lenguaje natural enfocados en la recuperación de información. Se presentan los conceptos referentes a dichas técnicas, así como los resultados de su aplicación de manera individual y a través de combinaciones entre ellas, mostrando así como se pueden realizar experimentos que involucren más de una técnica para mejorar sus resultados.

Una implementación de técnicas de PLN en el ámbito de la clasificación automática de textos se puede observar en [8]. En dicho trabajo se pretende utilizar la clasificación de documentos, implementando distintas técnicas de PLN para pre-procesar los textos, todo esto con el objetivo de recuperar información relevante de los textos estudiados. Además se presentan conceptos relevantes a dichos temas y un enfoque para la construcción de los experimentos de clasificación.

En [14] se propone la implementación de técnicas de máxima entropía para la clasificación automática de documentos. En ésta investigación, se utiliza la técnica de máxima entropía para clasificación de textos, estimando la distribución condicional de la clase dado el documento. Se realizan experimentos con varios conjuntos de datos y se compara la precisión del algoritmo a la obtenida usando un clasificador de Bayes ingenuo (Naïve Bayes). Obtienen como resultado que el rendimiento de la máxima entropía es significantemente mejor en la mayoría de los casos. Con esto concluyen que aún queda mucho trabajo en esta área, pero los resultados indican que la máxima entropía es una técnica prometedora para la clasificación automática de textos.

IV. METODOLOGÍA

La metodología seguida para completar la investigación, tiene como base las fases del descubrimiento de conocimiento en bases de datos (Knowledge Discovery in Databases, KDD), siendo éste un concepto muy utilizado en la actualidad en la extracción y análisis de conocimiento a partir de un conjunto de datos, tal como se indica en [16], el KDD se refiere al proceso completo de descubrir conocimiento útil a partir de datos. A pesar de que el KDD comprende un gran número de fases dentro su aplicación [16], en este trabajo se emplean las más relevantes para la investigación las cuales son:

A. Selección de los Datos

En esta etapa, se seleccionan de la base de datos de noticias, aquellas que serán usadas tanto en el conjunto de entrenamiento como el de pruebas dentro del proceso de clasificación, y se generan los documentos respectivos para cada conjunto.

Para este trabajo se hace una selección aleatoria de una misma cantidad de noticias por cada categoría con la intención de

mantener la uniformidad de los datos y no brindar mayor importancia a alguna de las categorías estudiadas.

B. Pre-procesamiento

Durante esta fase, se realiza la selección de las distintas técnicas que serán utilizadas para pre-procesar las noticias, tanto las contenidas en el conjunto de entrenamiento como en el de pruebas.

Los documentos que serán procesados son los generados en la fase previa, produciendo internamente nuevos documentos de entrenamiento y pruebas durante la ejecución de cada experimento realizado.

C. Clasificación

Se realiza la categorización automática de las noticias, esto por cada una de las combinaciones de técnicas seleccionadas en la etapa anterior, primero se entrena el clasificador con el archivo de noticias de entrenamiento, generando así el modelo de clasificación.

En cuanto a la presente investigación, dicho modelo consiste de un archivo binario, generado por el método de entrenamiento del algoritmo de clasificación utilizado. Este modelo es usado seguidamente en la fase de pruebas, haciendo uso del respectivo archivo con las noticias pertenecientes al conjunto de pruebas y usando dicho modelo para la clasificación.

D. Análisis

Se toman los resultados obtenidos durante la etapa de clasificación y se realiza un análisis comparativo entre cada uno de ellos, el objetivo de esto es determinar cuáles combinaciones de técnicas producen mejoras en los resultados de clasificación al ser usadas en la etapa de pre-procesamiento, teniendo como referencia un experimento ejecutado sin su utilización.

V. Fase Experimental

En los experimentos realizados se utilizó un cuerpo de noticias provenientes de una base de datos, la cual contiene aproximadamente 2500 noticias por cada una de las siete categorías: deportes, economía, mundo, política, salud, sucesos y tecnología; tales textos son correspondientes a la versión web del diario El Nacional.

Dicha fuente se presenta como un conjunto de datos atractivo para el estudio ya que es un diario de gran circulación a nivel nacional, es de fácil acceso y tiene un repositorio de noticias abundante que cubren la variedad de categorías estudiadas.

Estas noticias representan la parte central de los experimentos, siendo la variación de las cantidades utilizadas para entrenamiento y pruebas un factor común en cada experimento configurado. Los lapsos de tiempo entre los cuales están comprendidas las noticias por categoría se presentan en la Tabla I.

Se toma en cuenta la cantidad de noticias necesarias para completar el total de 2500 noticias por categoría, las cuales no tienen el mismo volumen por lapso de tiempo, es por esta razón que fue necesario tomar espacios de tiempo independientes por categoría en estudio.

Tabla I: Espacio de Tiempo por Categoría de las Noticias Almacenadas

Categoría	Desde	Hasta
Política	25/11/2015	03/02/2016
Economía	28/11/2014	03/02/2016
Mundo	03/08/2015	03/02/2016
Deportes	22/08/2015	03/02/2016
Sucesos	28/07/2015	04/02/2016
Salud	12/04/2013	04/02/2016
Tecnología	27/01/2014	04/02/2016

Para la clasificación se hace uso del principio de máxima entropía enfocado en la clasificación de textos, este algoritmo es proporcionado por la librería OpenNLP de Apache [4], la cual presenta un conjunto variado de herramientas para el procesamiento de texto en lenguaje natural. Los parámetros de configuración para el clasificador son:

- *Iterations*: cantidad de iteraciones de entrenamiento que hará el clasificador antes de generar el modelo.
- *Cutoff*: número mínimo de ocasiones en las que una palabra debe aparecer en el conjunto de textos de entrenamiento para ser tomada en consideración en el modelo.

Las tareas de PLN utilizadas para procesar los textos antes de su clasificación son los anteriormente descritos: reconocimiento de nombres, provisto por OpenNLP [4]; *stemming*, implementado por medio del uso de *Snowball* [17], un software para procesamiento de texto el cual permite manejar el proceso de *stemming* para el idioma español; y eliminación de palabras vacías, utilizando un documento previamente creado con un listado de las palabras que serán eliminadas.

Por lo tanto, de acuerdo a estos componentes, las variables que se ven modificadas entre distintos experimentos son:

- Cantidad de noticias de entrenamiento y pruebas por cada categoría.
- Valores para los parámetros del clasificador, *iterations* y *cutoff*, los cuales son requeridos al momento de generar el modelo.
- Técnicas de PLN a utilizar y orden de ejecución de las mismas.

Para medir la efectividad de los experimentos se utiliza el Valor-F o Medida F con un parámetro $\beta = 1$, siendo esto también llamado F1. Esta decisión está motivada en que no se pretende dar una importancia mayor a la precisión o la cobertura, por lo tanto, se utiliza dicha medida para mostrar de forma equitativa la influencia de ambos parámetros en los resultados generales de los experimentos.

Debido a que el resultado deseado por parte de la investigación comprende la selección de una combinación de parámetros para una serie de variables definidas, es necesario determinar el mejor valor para cada una de dichos parámetros, esto se ha conseguido al realizar la experimentación atravesando las etapas que a continuación se describen:

A. *Selección de los Datos*

Esta etapa comprende la realización de diversos experimentos, con el objetivo de establecer una cantidad de noticias que ofrezca buenos resultados en las fases de entrenamiento y pruebas. El objetivo es conseguir una proporción de documentos que brinde un alto nivel de eficiencia en la clasificación, sin llegar a presenciarse un sobreajuste en los datos.

Se ha desarrollado una serie de experimentos con esta premisa para buscar el punto en el cuál se pueda obtener un buen nivel de eficiencia disminuyendo la posibilidad de sobreajuste. Para esto se han utilizado variaciones de 50 noticias de entrenamiento por cada una de las siete categorías utilizadas (política, economía, mundo, deportes, sucesos, salud y tecnología), y utilizando siempre 500 noticias para pruebas por cada una de ellas, manteniendo el resto de variables ajustándose a los valores por defecto del algoritmo de máxima entropía utilizada.

Cabe destacar que todos los valores presentados durante el desarrollo del trabajo, corresponden al valor de F1 promedio obtenido al probar cada experimento, sin hacer énfasis en una categoría en particular ya que el objetivo es hacer una comparativa experimental del rendimiento global de los métodos utilizados.

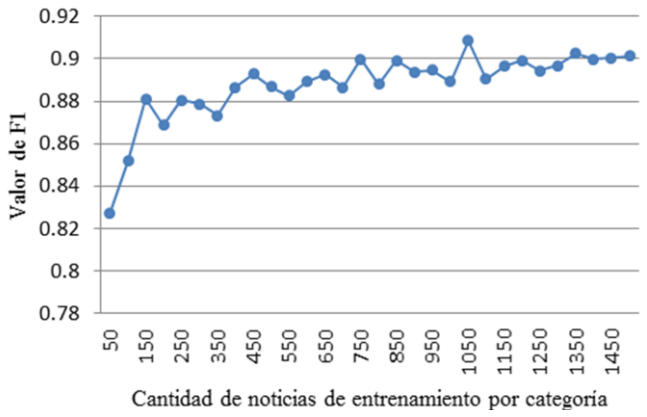

Figura 1: Variación de la Medida F1 con el Cambio en la Cantidad de Noticias de Entrenamiento por Categoría

En la Figura 1 se aprecia gráficamente el resultado de los experimentos descritos, se puede ver que inicialmente los valores van aumentando pero gradualmente tienden a alcanzar un cierto nivel de estabilidad luego de usar 1150 noticias de entrenamiento por categoría.

Nótese que la variación en los resultados luego de este punto no es muy significativa, a partir de aquí se considera el uso de entre 1150 y 1500 como rango de noticias de entrenamiento a usarse por cada uno de los experimentos que involucren modificación de técnicas a realizarse.

B. *Ajuste de Parámetros del Clasificador*

Seguidamente, se procede a definir valores para los parámetros base del clasificador, iteraciones y apariciones (*cutoff*).

Dado que en los trabajos revisados no se evidenció un análisis detallado de estos parámetros, se hace uso de los valores por defecto asignados por los desarrolladores del algoritmo como punto de partida, los cuales son: 100 iteraciones y un *cutoff* de 5; además se realizan variaciones independientes de dichos parámetros para verificar el impacto de cada uno de ellos en el resultado del clasificador.

Las variaciones sobre estos valores son realizados sobre los mismos experimentos ejecutados en la etapa anterior, cambiando en primer lugar el valor de iteraciones a 200, por otra parte el *cutoff* a 7, y en última instancia una mezcla de ambas variaciones, la comparación entre los resultados de éstos experimentos y el experimento base se observa en la Tabla II.

En la Tabla II, se presenta una comparación los valores de efectividad en término porcentual para el "estado estable" del clasificador en este experimento, se observa que en casi todos los casos los resultados son más altos usando 200 iteraciones y un *cutoff* de 5 para generar el modelo, alcanzando un valor promedio de F1 de 90,27% incrementando aproximadamente un 0,39% sobre la configuración predeterminada.

Por lo tanto, los experimentos creados a partir de este punto utilizarán 200 iteraciones y un *cutoff* de 5 como parámetros base para la generación del modelo.

Tabla II: Valores de F1 al Modificar Parámetros Base del Clasificador en el Estado Estable del Sistema

Cantidad de noticias de entrenamiento	**Variaciones de parámetros**			
	100 iteraciones / 5 cutoff	***200 iteraciones / 5 cutoff***	***100 iteraciones / 7 cutoff***	***200 iteraciones / 7 cutoff***
1150	89,6477	**90,1305**	89,3939	90,0186
1200	89,9158	90,353	90,0368	**90,3578**
1250	89,4282	**89,7412**	89,3981	89,7212
1300	89,6926	**90,0065**	89,6646	89,9205
1350	90,263	**90,9033**	90,1159	90,7889
1400	89,9722	90,3204	89,9171	**90,3521**
1450	90,0611	**90,239**	89,9144	90,0367
1500	90,1247	90,5267	90,0524	**90,6158**
Promedio	89,8881625	**90,277575**	89,81165	90,22645

C. *Pre-procesamiento y Clasificación*

Se realizan los experimentos correspondientes a una serie de combinaciones de las técnicas de PLN seleccionadas. En la Tabla III se listan las combinaciones de técnicas empleadas en experimentos, así como también el orden (representado numéricamente), en el que son ejecutados dentro de ellos, para posteriormente comparar sus resultados. Cada combinación es etiquetada con base en las técnicas que ejecuta y a su orden de aplicación; las siguientes etiquetas representan los nombres de las técnicas empleadas:

- N: Ninguna técnica.
- PVac: Eliminación de palabras vacías.
- RNom: Reconocimiento de nombres.
- Stem: *Stemming*.

La etiqueta "N" representa la ausencia de las tres técnicas en el experimento, marcando tal inexistencia con el texto "No

aplica"; la misma será utilizada como base de comparación para establecer la mejoría o empobrecimiento de los resultados obtenidos. Cada una de las combinaciones se probará en el estado de mayor estabilidad del clasificador.

Por ejemplo, la combinación RNom+Stem ejecutará en primer lugar el reconocimiento de nombres seguido del *stemming* (la eliminación de palabras vacías no se ejecuta ya que su columna indica el texto "No aplica"); en el caso de PVac+RNom+Stem se comienza ejecutando la eliminación de palabras vacías, posteriormente se utiliza el reconocimiento de nombres y finalmente el *stemming*.

Es necesario resaltar que debido a las propiedades del clasificador y buscando reducir el ruido en el análisis, al finalizar el procesamiento de los textos por cada combinación de técnicas, se eliminan los signos de puntuación y todas las palabras son transformadas a minúsculas.

Esto se debe a que los signos pueden causar ruido y una misma palabra se puede considerar en dos ocasiones si se presenta en mayúscula y minúscula, por ejemplo las palabas "Presidente" y "presidente" serían tomadas como dos entidades distintas, lo cual es solucionado al transformar todo el texto a letras minúsculas.

En la Sección VI se comparan los resultados obtenidos por cada una de las combinaciones usadas y se realiza un análisis detallado de los mismos, con la intención de obtener la configuración que brinde los mejores resultados al momento de clasificar documentos en el ámbito de estudio actual.

Tabla III: Combinaciones de Técnicas Usadas en los Experimentos

Combinación	**Técnicas aplicadas y orden**		
	Eliminación de palabras vacías	***Reconocimiento de nombres***	***Stemming***
N	No aplica	No aplica	No aplica
Stem	No aplica	No aplica	1
RNom	No aplica	1	No aplica
PVac	1	No aplica	No aplica
RNom+Stem	No aplica	1	2
PVac+Stem	1	No aplica	2
RNom+PVac	2	1	No aplica
PVac+RNom	1	2	No aplica
PVac+RNom+ Stem	1	2	3
RNom+PVac+ Stem	2	1	3

VI. Análisis de Resultados

En la Tabla IV se presentan los resultados obtenidos por cada combinación de técnicas de procesamiento de lenguaje natural probadas, al igual que antes los valores mostrados corresponden a la medida F1 en términos de porcentaje, la cual le asigna un peso igual a la precisión y cobertura, dando la posibilidad de evaluar ambas como un solo conjunto. Se pueden observar igualmente las combinaciones que brindan un mejor resultado por cada uno de los conjuntos de entrenamiento utilizados, así como el valor promedio de F1 para cada una de tales composiciones de técnicas.

Se puede observar en los resultados presentados, el incremento del valor promedio obtenido por las instancias que hacen uso de la eliminación de palabras vacías en comparación al resto. De hecho, al usar únicamente dicha técnica (columna *PVac*), se ha obtenido el segundo valor promedio más elevado con un F1 de 90,5863%, solamente por detrás de los experimentos bajo la combinación *PVac+Stem*, notándose así el impacto positivo de la eliminación de palabras vacías sobre el análisis.

Los experimentos realizados al utilizar únicamente el *stemming* (columna Stem), el reconocimiento de nombres (columna RNom) y la combinación de ambas (RNom+Stem), provocan una disminución en el valor medio de F1 con respecto al obtenido al promediar las instancias que no hacen uso de ninguna técnica. Sin embargo estos valores son mejorados al incorporar la eliminación de palabras vacías al análisis.

Un factor que puede influir en los resultados brindados por el clasificador, son las categorías de documentos tomadas en cuenta por el experimento y la correlación entre ellas. Si una clase guarda mucha relación con otra, varios de los documentos pertenecientes a ambas clases podrían ser asignados a la categoría incorrecta, un ejemplo de esto serían "política" y "economía", las cuales comúnmente están relacionadas principalmente en temas de carácter monetario.

Por lo tanto experimentos que se enfoquen en categorías con temas distanciados, como "política" y "deportes" tendrán mejores resultados con respecto a aquellos enfocados únicamente en clases con posibles correlaciones como "política" y "economía".

También se puede aumentar el error si se añaden categorías muy generales como "mundo", la cual presenta distintas noticias en diversas áreas siendo la única relación entre ellas su carácter internacional. Estos puntos se pueden observar en términos numéricos más adelante.

Al mismo tiempo se puede considerar que entre menor sea la cantidad de categorías estudiadas, se requerirá de menos documentos de entrenamiento por cada una para conseguir una efectividad elevada.

Esto se debe a que el clasificador necesitará hacer menos relaciones entre las entidades y las categorías, la misma línea de pensamiento se puede aplicar en el caso de que la cantidad de categorías estudiadas aumenten, situación en la cual el algoritmo requerirá de un número mayor de documentos de entrenamiento.

Siempre es necesario tener en consideración que en un entorno de clasificación automática de documentos, la calidad de los datos de entrenamiento es esencial. Por lo tanto se requiere tener una buena fuente de información para minimizar en lo posible el error de categorización ya que cualquier problema externo puede afectar también al clasificador, por ejemplo palabras mal escritas o el uso de documentos mal categorizados por la fuente en la fase de entrenamiento.

Habiendo considerado diversos factores que pueden afectar directamente los resultados de la clasificación, se prosigue con la selección de la configuración que brinda los mejores valores de F1 la cual será considerada como la propuesta para mejorar la tarea de categorización automática de documentos.

Tabla IV: Valor de F1 por cada Combinación de Técnicas de Procesamiento de Texto con Distintos Tamaños de Noticias de Entrenamiento

Noticias de entrenamiento	Combinación									
	N	***Stem***	***RNom***	***PVac***	***RNom+Stem***	***PVac+Stem***	***RNom+PVac***	***PVac+RNom***	***PVac+RNom+Stem***	***RNom+PVac+Stem***
1150	90,1305	90,2158	90,0817	90,4741	90,3085	**90,8743**	90,3705	90,2757	90,7568	90,6999
1200	90,353	90,7211	90,305	90,6105	90,5559	90,8211	90,3552	90,5495	90,9576	**91,0169**
1250	89,7412	89,6299	89,8577	**90,2848**	89,8389	90,1867	90,1694	90,202	90,1015	89,9291
1300	90,0065	89,78	89,9923	**90,4723**	89,569	90,1605	90,2841	90,2249	89,8604	90,0047
1350	90,9033	90,5394	90,5405	**90,9303**	90,691	90,9199	90,664	90,6665	90,6966	90,6115
1400	90,3204	90,5173	90,587	90,8141	90,3478	**91,0381**	90,4245	90,593	90,7567	90,6431
1450	90,239	90,1238	89,7342	90,5489	89,928	**90,5498**	89,9326	90,1836	90,2598	90,2691
1500	90,5267	89,8328	90,4647	90,5551	89,9779	90,2296	**90,7569**	90,6934	89,8856	89,8852
Promedio	90,2776	90,17	90,1954	90,5863	90,1522	**90,5975**	90,3697	90,4236	90,4094	90,3825

De la Tabla IV se extraen como los mejores resultados un 91,0381% obtenido al ejecutar la eliminación de palabras vacías seguida del *stemming* (*PVac+Stem*) en un entorno con 1400 noticias de entrenamiento por categoría, y un 91,0169% logrado al ejecutarse el reconocimiento de nombres, eliminación de palabras vacías y *stemming*, en ese orden (*RNom+PVac+Stem*), utilizando el conjunto de 1200 noticias de entrenamiento por cada categoría.

A simple vista, estos dos valores son prácticamente iguales, ambos representan una efectividad en la clasificación de 91%, por lo que cualquiera de las dos combinaciones podría ser tomada como la ideal, por lo tanto, se requiere un análisis más detallado de ambas opciones para determinar cuál de las dos debe ser seleccionada como la mejor.

En la Tabla V se presentan los resultados detallados de precisión, cobertura y F1 por cada categoría para la configuración correspondiente al usar la combinación *PVac+Stem* con 1400 noticias de entrenamiento por categoría. La misma clase de resultados se muestran en la Tabla VI para la combinación *RNom+PVac+Stem* con 1200 noticias por categoría para la fase de entrenamiento, de este modo se puede comparar con mayor detalle los valores de evaluación para estos experimentos. Al igual que en los casos anteriores los resultados son presentados en términos de porcentajes para todas las medidas de evaluación.

Tabla V: Valores de Evaluación Detallados al Usar la Combinación PVac+Stem con 1400 Noticias de Entrenamiento por Categoría

Categoría	Medida de evaluación		
	Precisión	***Cobertura***	***F1***
política	88,7129	89,6	89,1542
economía	89,4212	89,6	89,5105
mundo	87,2385	83,4	85,2761
deportes	96,6203	97,2	96,9093
sucesos	92,1606	96,4	94,2326
salud	93,2271	93,6	93,4132
tecnología	89,7541	87,6	88,664
Promedio	91,0192	91,0571	91,0381

Tabla VI: Valores de Evaluación Detallados al Usar la Combinación RNom+PVac+Stem con 1200 Noticias de Entrenamiento por Categoría

Categoría	Medida de evaluación		
	Precisión	***Cobertura***	***F1***
política	87,2624	91,8	89,4737
economía	89,6282	91,6	90,6034
mundo	85,1927	84	84,5921
deportes	95,9759	95,4	95,6871
sucesos	93,4263	93,8	93,6128
salud	93,5743	93,2	93,3868
tecnología	92,1776	87,2	89,6197

Categoría	Medida de evaluación		
	Precisión	***Cobertura***	***F1***
Promedio	91,0339	91	91,0169

En primer lugar, se puede notar que las categorías más distintivas o específicas brindan mejores resultados que otras con mayor correlación, esto se aprecia con "deportes", "sucesos" y "salud", las cuales presentan en ambos casos valores entre el 93 y el 96% mientras que otras como "política", "economía" y "tecnología" se acercan al 90% y "mundo", la menos específica de las categorías consideradas presenta los valores más bajos, alrededor de 85%.

Para verificar la igualdad de ambas configuraciones enunciada anteriormente, se empleó una prueba de hipótesis para comparación de medias con varianza conocida y muestras normales, o prueba Z cuya fórmula se enuncia en [18] y [19], utilizando los valores respectivos de este experimento como se muestra a continuación:

$$Z = \frac{91{,}0228 - 90{,}9965}{\sqrt{\frac{15{,}8392}{7} + \frac{13{,}3243}{7}}} = \frac{0{,}0263}{2{,}0411} = 0{,}0129 \approx 0{,}01 \quad (2)$$

La hipótesis nula (H_0) planteada corresponde a la igualdad de ambas medias, significando esto que ambos métodos son equivalentes; mientras que la hipótesis alternativa (H_1) equivale a que dichos valores son diferentes siendo de la misma manera distintas ambas combinaciones.

Siendo la regla de decisión rechazar H_0 si:

$$|z| > z_{\alpha/2} \quad (3)$$

Para la prueba, se ha tomado un valor de $\alpha = 0{,}05$, siendo entonces el punto crítico $z_{\alpha/2} = 1{,}96$. Entonces, debido a que $Z = 0{,}01$ es menor que $z_{\alpha/2} = 0{,}025$, se encuentra dentro de la zona de aceptación de la hipótesis nula para el nivel de significancia de 0,05 por lo que se acepta H_0.

Por lo tanto, dado que dicha hipótesis establece que ambas medias son iguales, se puede concluir que no se han encontrado diferencias estadísticamente significativas entre ambas medias, o análogamente, no existe evidencia que demuestre que las medias de ambas muestras sean diferentes. Este argumento se asegura con un nivel de confiabilidad de 99,2% de acuerdo al valor-p calculado en la Ecuación (4).

$$p = P(|Z| > 2{,}16) = 0{,}992 \quad (4)$$

A causa de esto, se puede requerir de un mayor grado de subjetividad por parte del analista al momento de decidir cuál de las dos combinaciones será seleccionada por sobre la otra.

Para determinar cuál configuración se ajusta mejor al enfoque de un investigador, se pueden considerar los siguientes criterios adicionales:

A. Resultados para una Clase Específica

Puede que, por algún motivo, el interés del analista o el usuario interesado en la información sea mayor para una categoría en especial, por lo tanto, si en este caso se le da mayor importancia a "economía", la elección recaería en utilizar la combinación *RNom+PVac+Stem* aplicada en un entorno con 1200 noticias de entrenamiento por categoría, ya que presenta un valor de F1 de 90,6% a comparación del 89,51% presentado por la otra opción.

Sin embargo, si se considera más importante la categoría "deportes" debería ser seleccionada la combinación *PVac+Stem* con 1400 noticias de entrenamiento por cada categoría ya que brinda un F1 de 96,9% frente al 95,68% presentado por la segunda alternativa.

B. Cantidad de Información de Entrenamiento

La base de datos utilizada contiene alrededor de 2500 noticias disponibles por cada una de las categorías estudiadas. La primera de las configuraciones comparadas utiliza 1400 noticias de entrenamiento por categoría, esto corresponde a un valor cercano al 56% del total de noticias, mientras que el segundo objeto de estudio presenta un entorno con 1200 noticias de entrenamiento por cada categoría, siendo esto alrededor del 48% del total disponible.

Por lo tanto esta última debería ser la configuración seleccionada bajo este criterio, ya que requiere de una menor cantidad de documentos de entrenamiento para alcanzar un valor alto de F1, evitando así de la mejor manera la probabilidad de un sobreajuste durante el entrenamiento del clasificador.

C. Complejidad del Pre-procesamiento

Si se considera que entre más técnicas se utilicen para procesar el texto antes de pasarlo al clasificador (sea en fase de entrenamiento o pruebas), mayor será el tiempo necesario para clasificar los documentos deseados y obtener los resultados del proceso, en situaciones como la que se presenta lo más natural sería seleccionar aquella configuración que utilice la menor cantidad de técnicas posible.

En este caso sería seleccionada la primera de las dos configuraciones utilizadas ya que además de utilizar menos técnicas (dos contra tres), hace uso de las menos complejas en cuanto a requerimientos de procesamiento.

D. Media del Resto de Experimentos para la Combinación

Si se desea tomar en consideración los otros experimentos, con cantidad de noticias de entrenamiento distintas, para cada combinación se puede obtener un valor promedio de F1.

La primera configuración utiliza la combinación *PVac+Stem* la cual, como se muestra en la Tabla IV tiene un valor promedio de 90,5975% mientras que la segunda hace uso de la combinación *RNom+PVac+Stem*, esta presenta un resultado medio de 90,3825% siendo 0,215% peor que la anterior, por lo tanto en este caso sería seleccionada la primera configuración ya que presenta mejores resultados de forma general.

Al tomar en consideración cada uno de los puntos anteriores, sin darle mayor importancia a una categoría sobre otra, se considera que la mejor elección corresponde a la primera configuración comparada, ya que presenta una mejor media general utilizando un procesamiento del texto menos extenso. Por lo tanto, para el caso de estudio actual la configuración completa de los parámetros requeridos es la presentada en la Tabla VII.

Tabla VII: Configuración Utilizada para Obtener el Mejor Resultado en la Clasificación

Variable	Valor
Iteraciones	200
Cutoff	5
Porcentaje de noticias de entrenamiento por categoría	56%
Técnica de PLN #1	Eliminación de palabras vacías
Técnica de PLN #2	*Stemming*

Es necesario recalcar que los resultados son afectados directamente por la calidad de la información contenida en los documentos, ya que las técnicas de PLN mejoran o empeoran los valores dependiendo de la información que deben procesar. De la misma manera el clasificador, en este caso de máxima entropía, también depende completamente de los datos de entrada.

Es importante señalar que en este trabajo se utilizaron los parámetros ajustados a una prueba, los cuales no aplican necesariamente para todos los contextos. Por lo tanto, con un conjunto de datos de entrada diferentes, puede existir un conjunto de parámetros más ajustados a dicho cambio, presentándose finalmente esta investigación como antecedente y base para futuras investigaciones.

VII. CONCLUSIONES

La cantidad de información usada en la fase de entrenamiento es un factor de vital importancia en un sistema de clasificación automática de documentos, por lo tanto, es necesario encontrar y establecer un tamaño de datos de entrenamiento que logre una eficacia elevada evitando al mismo tiempo la posibilidad de sobreajuste.

Cada técnica de procesamiento de lenguaje natural tiene un efecto distinto sobre los textos, su efectividad puede variar dependiendo del entorno de información sobre el cual es aplicada, por lo tanto es necesario probar distintas técnicas y combinaciones de las mismas, verificando cuál de ellas es capaz de brindar el mayor incremento en la eficacia de la clasificación para el contexto estudiado.

Debido a que el clasificador depende totalmente de los datos etiquetados de entrenamiento y, además de esto, toma como principio el no asumir nada usando la uniformidad en las probabilidades, es necesario que las categorías estudiadas y la información a clasificar sea lo suficientemente específica o bien pre-procesada para producir resultados de mayor calidad, mostrando así la necesidad de una buena selección de

parámetros y procesos de PLN antes de empezar a clasificar los documentos.

Debido a las características de los métodos estudiados, es importante resaltar que los cambios realizados sobre cualquiera de las variables presentadas, pueden generar diversos cambios sobre los resultados de la clasificación.

Al concluir la clasificación, es tarea del analista seleccionar la combinación de parámetros y técnicas que mejor se ajuste a sus criterios de selección, ya que estos pueden variar de acuerdo al ámbito en el cual se realizan los experimentos, por lo que un análisis detallado de los resultados obtenidos será necesario constantemente.

VIII. Trabajo Futuro y Aportes

Con la presente investigación se ha conseguido obtener un método alternativo para la clasificación automática de documentos, el cual puede ser usado como base para futuros trabajos en dicho ámbito de investigación.

La mayor parte de antecedentes investigativos relevantes a la categorización automática de documentos se enfocan en textos en inglés, por lo que este trabajo muestra la efectividad de los métodos utilizados en cuerpos de texto escritos en el idioma español.

Ya que en la presente investigación solo se hizo una variación sobre la cantidad de iteraciones y el *cutoff*, es factible probar de forma más exhaustiva las variaciones de estos parámetros con la intención de encontrar la combinación de valores que mejor se ajuste al entorno de investigación estudiado.

El porcentaje de documentos aplicados para entrenamiento no es necesariamente igual en todos los entornos, para otros ámbitos se podría obtener un valor mayor o menor al 56% propuesto en esta investigación, por lo que es recomendable partir de este valor y probar distintas disminuciones e incrementos del mismo con la intención de encontrar el mejor ajuste para el entorno de textos estudiados.

Si bien solo se han probado tres técnicas de procesamiento de lenguaje natural, se ha podido verificar que la eliminación de palabras vacías es un primer paso al momento de procesar texto para su posterior análisis.

De igual manera, es altamente recomendable probar otras técnicas además de las usadas en esta investigación, pues cada una tiene un efecto diferente sobre el texto y no es totalmente posible determinar cual dará resultados positivos o negativos en el ámbito estudiado.

Algunas técnicas que podrían ser tomadas en consideración además de las que se han manejado durante esta investigación son: etiquetado gramatical, para realizar una desambiguación del significado de una palabra en un contexto; resolución de correferencias, con la intención de descubrir relaciones directas entre las entidades previamente encontradas en el texto; lematización, cuyo objetivo es procesar un grupo de inflexiones para obtener una única palabra o lema que las represente a todas.

Finalmente es necesario destacar que todo el procedimiento correspondiente a la creación de los experimentos, la selección y variación de los valores para cada uno de los parámetros utilizados, así como la determinación de los mejores valores en cada etapa, es realizado de forma manual. Por lo tanto, un avance altamente considerable e interesante, es la inclusión de tareas de inteligencia artificial orientadas a la selección de los valores óptimos para cada una de las variables estudiadas. De esta manera el algoritmo se encargaría de realizar las variaciones en los valores y ejecutar los experimentos para finalmente determinar la mejor configuración para el ámbito de investigación estudiado, haciendo una sintonización de parámetros de forma automática.

Referencias

[1] R. Eíto Brun and J. A. Senso, *Minería Textual*, El profesional de la información, pp. 11-27, 2004.

[2] S. Vijayarani, M. J. Ilamathi, and M. Nithya, *Preprocessing Techniques for Text Mining-An Overview*, International Journal of Computer Science & Communication Networks, vol. 5, no. 1, pp. 7-16, 2015.

[3] A. C. Vásquez, H. V. Huerta, and J. P. Quispe, *Procesamiento de Lenguaje Natural*, Revista de Investigación de Sistemas e Informática, vol. 6, no. 2, pp. 45-54, 2009.

[4] Apache OpenNLP. https://opennlp.apache.org

[5] A. F. Anta, L. N. Chiroque, P. Morere, and A. Santos, *Sentiment Analysis and Topic Detection of Spanish Tweets: A Comparative Study of NLP Techniques*, Procesamiento del Lenguaje Natural, no. 50, pp. 45-52, 2013.

[6] P. G. Otero and M. G. González, *Técnicas de Procesamiento del Lenguaje Natural en la Recuperación de Información*, Novática, no. 219, pp. 42-47, 2012.

[7] R. Feldman and J. Sanger, *The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data*. New York: Cambridge University Press, 2007.

[8] M. A. P. Abelleira and C. A. Cardoso, *Minería de Texto para la Categorización Automática de Documentos*, Cuadernos de la Facultad, Universidad Católica de Salta, no. 5, pp. 11-45, 2010.

[9] K. Nalini and L. J. Sheela, *Survey on Text Classification*, International Journal of Innovative Research in Advanced Engineering (IJIRAE), vol. 1, no. 6, pp. 412-417, Julio 2014.

[10] V. Gupta and G. S. Lehal, *A Survey of Text Mining Techniques and Applications*, Journal of Emerging Technologies in Web Intelligence, vol. 1, no. 1, pp. 60-76, Agosto 2009.

[11] C. Goller, J. Löning, T. Will, and W. Wolff, *Automatic Document Classification: A Thorough Evaluation of Various Methods*, 2000.

[12] A. G. Ramírez de la Rosa, *Clasificación Automática de Resumenes de Tesis Basada en Algoritmos de Agrupamiento Jerarquicos*, Universidad Tecnológica de la Mixteca, Oaxaca, México, Tesis de pregrado 2008.

[13] M. Emms and S. Luz, *Machine Learning for Natural Language Processing*, European Summer School of Logic, Language and Information, Dublin, Irlanda, 2011.

[14] K. Nigam, J. Lafferty, and A. McCallum, *Using Maximum Entropy for Text Classification*, in IJCAI-99 Workshop on Machine Learning for Information Filtering, pp. 61-67, 1999.

[15] A. Téllez, *Extracción de Información con Algoritmos de Clasificación*, INAOE, Tonatzintla, Tesis de maestría 2005.

[16] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, *From Data Mining to Knowledge Discovery in Databases*, AI magazine, 1996.

[17] Snowball. http://snowball.tartarus.org

[18] R. E. Walpole, R. H. Myers, S. L. Myers, and K. Ye, *Probabilidad y Estadística para Ingeniería y Ciencias*, Octava edición, Pearson Educación.

[19] T. Bartz-Beielstein, *Experimental Research in Evolutionary Computation*, Springer, 2006.

Revista Venezolana de Computación
ISSN: 2244-7040
http://www.svc.net.ve/revecom
Vol. 4, No. 1, pp. 10-21, Junio 2017
Fecha recepción: 29/04/2017, Fecha aceptación: 30/07/2017

Experience on Software Product Line Domain Engineering for Mobile Computing

Francisca Losavio[1], Oscar Ordaz[1,2]
francislosavio@gmail.com, oscarordaz55@gmail.com

[1] Escuela de Computación, Laboratorio MoST, Universidad Central de Venezuela, Caracas, Venezuela
[2] Escuela de Matemática, Laboratorio MoST, Universidad Central de Venezuela, Caracas, Venezuela

Abstract: Domain Engineering (DE) is crucial to determine limits and feasibility of a Software Product Line (SPL), or family of similar products or systems sharing common and reusable elements or core assets in a domain or market sector. The main goal of this work is to present an industrial experience centered on the PLScope phase of DE, considering a Quality-driven Domain Engineering Process (Q-DEP), using a bottom-up strategy based on the study of the enterprise existing products, to reduce the effort in the subsequent DE phases. This approach is more low cost and light weighted than the proactive (top-down) approach; however, both are recommended for SPL development by the ISO/IEC 26550 and Software Engineering Institute frameworks. Q-DEP has been applied successfully on a small-medium size Mobile Computing (MC) consortium that wanted to migrate to SPL. However, SPL development requires a huge engineering effort to build its main reusable artifact, the Reference Architecture (RA), or instantiable schema to derive concrete SPL products. RA is a highly abstract software architecture defined by components and connectors, holding placeholders to perform instantiation to derive new products. MC is facing increasing software demand, being a development based on information technology, fast multimedia transmission via computer or any wireless connected mobile device. MC does not deal with complex systems, but it must withstand very fast development/delivery issues and priority quality requirements such as efficiency, portability, usability and availability. Developers' teams work independently and loose effort in programming resources. The MC consortium wanted to build a first asset repository, to start migrating to SPL; construction of RA was not planned at the project start. However, the domain and existing products study was crucial to build the first asset repository, and our bottom-up process reduced the effort of the subsequent phases, hence RA was relatively easy to build.

Keywords: Domain Engineering; Software Product Line (SPL); Mobile Computing; Reference Architecture; Asset Repository; Q-DEP; bottom-up strategy

I. INTRODUCTION

Software Product Lines (SPL) for a particular domain or market sector, is an approach to industrial software development that provides massive personalization of individual solutions, sharing elements from a repository of reusable software assets, organized into a *Reference Architecture (RA)* with an instantiable schema, to derive concrete products of the SPL family [1][2][3][4][5]. A *domain* is defined in [6] as the minimal set of properties describing precisely a family of problems in which a computational application or system is involved for their solution. *SPL Engineering (SPLE)* is a huge and costly process since it involves the construction of an evolutionary RA. Two main lifecycles drive this "heavy" *SPLE* process*: Domain Engineering (DE*), where the *RA* and the *Asset Repository* are constructed, and *Application Engineering (AE*), where the RA schema or *variability model* is instantiated to derive new products [5] and the Asset Repository is also updated. In this work, DE and its first phase, the *Product Line Scoping (PLScop)* as recommended in the new standard ISO/IEC 26550 [5] will be considered. It takes into account the study of existing products built by the enterprise or available on the market, which is the base of our RA *extractive bottom-up* development strategy, derived from architectural-centric approaches. The Software Engineering Institute (SEI) [31] mentions that the reactive/extractive approach has the advantage of a much lower cost to start an SPL migration because the core asset base is not built up-front, and an existing product can be extended with new features to incrementally get the RA; it is much used in industrial practice and it contributes to reduce the huge effort required by the subsequent DE phases as it has been observed in [17]. The reactive approach considers one existing product that is extended to conform a SPL; the extractive approach instead considers a refactoring of several similar existing products to identify common and variant components to achieve the RA. The PLScop phase [5] deals with a broad capture of domain knowledge, to delimit the SPL application ambit, including risk analysis, economical feasibility and identification of the SPL family of products to be constructed. PLScop becomes necessary, since the knowledge on the SPL domain should be captured early, to

reduce the major effort concentrated on the subsequent phases of *Domain Requirements Engineering (DRE)* and *Domain Design (DD)*, where RA is actually built. Quality assurance should also be taken into account early, because quality related to domain *functional (FR)* and *non-functional requirements (NFR)* is responsible in a major degree of the SPL variability; however most of the approaches leave it to the DD phase. Our basic idea is to consider software components, often-common components that respond to precise quality goals, provided by others, often-variant components. Traceability of *functional (FR)* and *non- functional (NFR)* is achieved by construction, thus guaranteeing the RA evolution.

The main goal of this paper is two-folded; on one hand an industrial experience in the *Mobile Computing (MC)* domain, applying successfully a quality driven "semi-agile" light weighted DE process is related. On the other hand, the complete process that has been defined, applied and illustrated with the industrial case study is presented.

The fast world-wide network connection responsible of the so called "globalization", gave origin to MC software development [7][8], an approach highly based on *Information Technology (IT)*, allowing fast multimedia (data, voice, video) transmission via a computer or any wireless connected mobile device. Mobile software development involves widespread use of IT tools, often free and open-source support platforms, toolkits, etc., that can change rapidly over time. The new *Mobile Software Engineering (MSE)* discipline emerges [8], with the main trends of mobile communication, mobile software and hardware. In particular, mobile software is characterized by: - *Web Applications (Web Apps)* displayed on the user's device through a browser and executed on a remote server, - *Native Applications (Apps)* designed and developed for a particular device, but downloaded and installed by the user on his device, and – *Hybrid applications*, exposing Web Apps contents in Apps formats. Mobile software is not intensive software and does not involve thousands of lines of code, but it must be developed very fast to satisfy a huge market demand and its dynamic nature. Moreover, it must have a certain quality level, expressed by quality requirements such as usability, efficiency and availability, being this crucial for the immediate acceptance or rejection of the product; the wide spread use of rapidly implemented and evolutionary IT characterizes this kind of development, being client satisfaction one of the main goals. However, the success of a software application, mobile or not, depends not only on its functional suitability representing adequate FR, but also on the satisfaction of the quality properties representing NFR, required by these functionalities to have a suitable behavior [8][9][10].

Good practices of software engineering, as recommended in [11] are not widespread in MSE [8]. Important aspects to be considered relative to the quality properties already mentioned, are: - the integration of hardware and software, - limited resources such as data storage and display for the *User Interface (UI)* on reduced size screens with different resolutions and devices, - frequent changes in IT, - use of interoperability standards, and traditional quality requirements, such as – portability to different platforms, - reliability w.r.t. availability of Internet connection, - security to guarantee access control, and finally - efficiency in the UI display and process services to execute functionality. In view of the growing demand of mobile applications from 2007[1] with the iPhone success, a survey in [8] on main practices used in mobile software development, reported the following important aspects:

1. Applications are "small", few thousands of lines of code in average, involving only one or two developers.
2. There is quite a difference between Apps and Web Apps.
3. No development methods are used in this domain.
4. Developments are not documented and very few metrics are registered.

Notice that points 3 and 4 are against software engineering best practices[2]. Nevertheless, lots of commercial platforms are available to develop mobile applications, and as it will be shown later on, they allow guaranteeing to a certain extent the accomplishment of priority quality properties required by the functionalities. With respect to point 3, the so called "agile" methods, practices and techniques [12][13][14], usually driven towards a rapid UI development to achieve client satisfaction, are recommended for MSE [8][15].

We faced the problem of a small-medium sized (more than 20 employees) mobile software enterprise that wanted to identify reusable software assets to build a first repository of reusable software components (modules, toolkits, APIs, mechanisms, etc.). They were working on three different sub-domains of MC (financial transactions, healthcare systems and entertainment contents), and the programming effort was triplicated, since it was not known which semantically similar components were actually used in which sub-domain. Our proposed solution was a "slow" migration towards a MC product line, starting with the constructions of a high-level Core Asset Repository, holding reusable and variant components used within the different products developed by each sub-domain of the enterprise. The project at first did not pretend to construct a RA for the enterprise SPL and only the PLScop phase was to de developed. However, in order to construct a first draft of the repository, a complete DE process had to be done, but they had no time and no human resources, dedicated mostly to development, to employ into such a huge task. In consequence, a "light" or "semi-agile" DE phase of SPLE was proposed, including *PLScop*, *Domain Requirements Engineering (DRE)* and *Domain Design (DD)* phases [5]. However, for the product line migration to be successful, the architecture and other core assets must be robust, extensible, and appropriate to future product line needs [31], and a RA could be constructed in this way for each sub-domain, profiting from the extractive bottom-up design. The process should start by reengineering the available information to construct the architecture of each product/system within the sub-domain, studying similarities and differences among components on the basis of interviews and meetings (architecture documentation was absent, as usual in industrial practice). Based on the extractive strategy, the bottom-up approach was applied, meaning that more than one product built by the enterprise had been considered, to construct automatically an initial *Candidate Architecture (CA)* represented by a connected graph [10], by the graph union of the existing products' architectures,

[1] https://wikipedia.org/wiki/History_of_iPhone
[2] http://technav.ieee.org/tag/4655/best-practices

designed at a high abstraction level, also represented by graphs. To achieve this, the process *Q-DEP: Quality-oriented Domain Engineering Process* was defined, adapted from [24], and applied during a period of six months.

Q-DEP concerns the main phases of the DE lifecycle including a first assessment for an Asset Repository; it does not involve source code reengineering; the information on the products developed by the enterprise was captured via an "agile" practice, from a quiz and several interviews with the developer teams' leaders, until a consensus on the configuration of the existing products' architectures, in terms of high-level description of components and connectors [16], is achieved. Notice that similarities among the products developed are assumed to exist because they belong to the same domain/sub-domain. We recall that in general industrial practice the logic view of the software architecture is not developed, and often the only documentation available are deployment diagrams representing physical nodes where components are running [32]; however, it is required to construct the RA in the SPLE approach, to identify *common components* and *variant components* (they do not appear in all products) involved. Fine grained goals of this work are: a) design an RA for each sub-domain, according to a bottom-up approach based on the refactoring of existing products' available documentation, built and used by the enterprise, b) conform a first draft of the assets, a list of common and variant components that are used within the enterprise products, and c) Illustrate the stepwise application of the process considering only one of the sub-domain for this presentation.

In addition to this introduction, this work is structured as follows: Section II discusses some related works; Section III describes the Q-DEP process (phases, activities); Section IV shows Q-DEP applied to the Healthcare Information Systems (HIS) case study of the sub-domain of the *XX-MC* enterprise. Finally the conclusion and perspectives are presented.

II. Related Works

Very few scientific research works were found discussing MC and even less about SPL for mobile computing. Few use a bottom-up strategy, which we consider a much more pragmatic, fast and practical approach to SPL development, since the fact of having one or more existing products available on the market or developed within the enterprise, is a more common situation than to build the SPL from scratch (top-down approach) [10][31]. The new standard SPL Reference Model [5], even favoring a global proactive top-down approach, introduces explicitly an initial scoping phase to handle this problem, including a product portfolio that can be built from products on the marketplace and/or from the enterprise own products, using bottom-up techniques. In this work, we have adopted this strategy, also to reduce the effort in the subsequent DE phases, including however quality issues, even in this first phase, differing from [5], which delegates these activities to the late DD phase [17]. The following works will be discussed:

- A method is proposed in [18] for UI development in the MC domain, called GeMMINi; *Model Driven Design (MDD)* is used to model transformations, combined with feature modeling [19]. The method is very informally specified; it describes at a high abstraction level UI requirements and the device variants, applying model transformations and generating code to obtain Apps for UI on different devices. A catalogue of patterns is defined to translate UI abstract concepts into concrete device specifications; it is used to configure the transformations; the user interaction is specified into "units" with UML class diagrams, containing the data structure descriptions. The specification of properties and variants of the devices are provided by an ontology-based feature model [19][20], to specify some aspects of the interaction. As it is usual in feature models approaches, no RA is considered.

  This paper involves the DE and IA cycles of SPLE; however, just the UI component is treated, and recent mobile computing development toolkits already exist to conform the UI for each device, such as IONIC[3]. Only functional variability is handled, because the usual feature model only considers this aspect, even if lots of works have been done to include also non- functional variability [10]. Our approach with Q-DEP is more fine-grained; we handle the refactoring of all main components obtained from the architectural configurations of similar products within the enterprise. We don't use feature modeling nor MDD, using a scenario-based approach instead [21]; an initial CA is automatically constructed, by the "union" of the graphs representing the architectures of the existing products, establishing a first variability model that can be completed with additional information. We retrieve traceability among FR and NFR on the bases of tables representing scenarios [21], where each component or service solving a quality property required by a functionality, is specified.

Previous works [10][22][23][24] have evolved to define the present Q-DEP process. They will be discussed in what follows:

- The work in [22] treats the robotics domain. Reengineering techniques are mostly used, from code or documentation, to reconstruct the architecture of existing products in the enterprise. RA is built manually and directly, without considering an intermediate architecture. The justification of the satisfaction of NFR is very informal and standards are not used to specify quality properties. It has inspired our present research.
- The idea of representing a software architecture [16] by a connected graph was developed in [23], to perform the automatic "union" of the refactored architectures of similar existing products (belonging to the same domain), according to a bottom-up strategy inspired in [22]; in this way a first CA was constructed, showing common and variant components obtained from the products considered. Quality properties are grouped as scenarios in the Extended Quality Model (EQM) Table, and were specified by the standard ISO/IEC 25010 quality model [9]; EQM contains components and their required quality properties; however traceability among FR and NFR was not completely justified and the variability model was established according to an optimization process. The process was applied to the robotics domain, as in [22].

[3] https://ionicframework.com/docs

- The process originally defined in [21] was reformulated in [10] into a new process NF-VAR, to construct RA, combining the extractive bottom-up strategy with goal-oriented techniques [25], using the *Softgoals Interdependence Graph (SIG)* diagram, to complete the CA obtained automatically [23], with new components introduced to satisfy NFR. NF-VAR was applied to the Healthcare Information Systems (HIS) domain, using three open-source market products, OpenEMR[4], PatientOS[5] and Care 2X[6]. However, the use of the SIG was not straightforward, even if tools such as GRL[7] were available; it is difficult to handle complexity with this diagram, which does not offer a standard notation.
- The *QuaDRA (Quality-oriented Design of Reference Architecture)* process was specified for SPL in [24]; the new ISO/IEC 26550 standard defining the SPLE Reference Model [5] was followed to include PLScop, the first DE phase, to reduce the effort in the subsequent DRE and DD phases of DE. The NF-VAR process defined in [10] fitted well into PLScop, to construct the SPL product portfolio and to produce automatically CA and the EQM Table; the Domain Scoping phase taken from [26], was used to complete CA with additional information captured by different stakeholders viewpoints, including quality as a new intrinsic facet to describe stakeholders viewpoints. In this way, business processes specified in BPMN were integrated to the process, attaching quality requirements to each functional task. In this way the use of the SIG was avoided.

  Q-DEP was adapted from [24]: business processes specification were not included to have a more "light weighted" DE process. Requirements elicitation was done using "agile" practices by interviewing stakeholders and performing meetings to achieve a consensus on components and connectors conforming the architectural configuration [16]. Our process is highly based on the provide/require scenarios tables for quality properties traceability, maintaining the CA automatic construction by a bottom-up strategy.

III. Q-DEP: Quality-driven Domain Engineering Process

A PL Scoping phase including Domain Analysis activity and some technics inspired from agile methodologies are focused in Q-DEP to achieve a "light" DE process; it does not consider business processes as in [24], being more oriented towards a "naïve" but practical approach of DE, involving direct interaction with project leaders. Three main phases of the DE lifecycle have been considered: *1. PLScop (SCOP)* with main activities: – Build Product Portfolio (ProdPort), - Agile Domain Analysis (ADA), - Assets Identification (AssetT), and Build Global Assets (GLAsset); *2. Domain Requirements Engineering (DRE)* with main activities: – Build Candidate Architecture (CA), Build the UML representation of CA, and Build EQM, and - *3. Domain Design (DD)* with main activities: – Build Reference Architecture (RA), – Elaborate General Assessment Document (GAD). The complete process is specified in Figures 1 and 2; the main effort is concentrated in Phase 1 SCOP, but this will reduce the work in the subsequent phases. When ADA is mentioned [27], we do not pretend to develop or use a complete "agile" methodology [12] [13][14], but just use some basic technics such as interviews and meetings with main project leaders. We want to achieve a consensus on the enterprise product components, connectors and architectural solutions. The number of meetings will depend on the complexity observed during the first meeting to elicitate the information on the enterprise domain/sub-domain(s). We recall that even if agile methods are suggested for the MC domain, we face an SPL context where a family of similar systems is built, and agile methods seems to be better suited for single systems' development [28].

To achieve a *variability model* imbedded into the RA [5] showing the placeholders to be intatiated, the products' *common components* are identified from a semantic viewpoint, considering the accomplishment of functionally similar tasks; the *variants components* will then be glued also according to the similarity of their tasks, into categories called *variation points* [3], which are sets of variant components.

IV. Application of Q-DEP to the Healthcare Information Systems Sub-domain

Q-DEP was applied successively to three sub-domains (financial transactions, healthcare information systems and entertainment contents) of the XX-MC enterprise. Recall that the enterprise problem was to determine the main components, modules, support platforms and/or toolkits that had been used in product developments, to avoid repeated programming efforts. The application of the process was done for the three sub-domains during 6 months; about a hundred components were identified as assets for the three sub-domains.

Q-DEP will be applied here to the Healthcare Information Systems (HIS) sub-domain.

The general HIS architecture is a hybrid event-based style, SOA[8]/Layers, following a client-server model for distribution and communication. HIS must facilitate transparent sharing of different kinds of medical information such as EHR and laboratory and imaging results, offering also telemedicine services that can be performed on-line at remote locations, with wide support of information technology. The use of standards such as HL7[9], HL7 CDA, LOINC[10], and DICOM[11] are mandatory for interoperability of HER, and laboratory and imaging results. Nevertheless, in actual medical practice, SPL for HIS have not yet been completely defined, developed and adopted; the lack of agreement on medical standards and psychosocial issues makes difficult the interoperability of EHR, and HIS general adoption is still difficult, even if specific laws and regulations towards these goals have been promulgated worldwide.

[4] https://www.open-emr.org
[5] https://sourceforge.net/projects/patientos
[6] https://www.care2x.org
[7] Goal-oriented Requirement Language, https://www.cs.toronto.edu/km/GRL
[8] Service-Oriented Architecture
[9] https://www.hl7.org
[10] Logical Observation Identifiers Names and Codes
[11] Digital Communication in Medicine

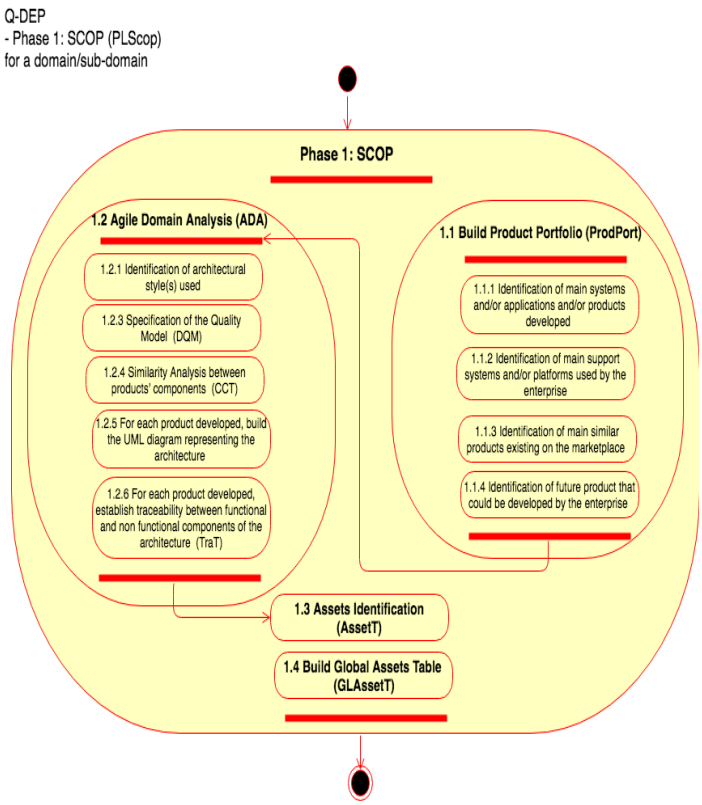

Figure 1: Q-DEP – Phase 1 SCOP – UML[12] [29] Activity Diagram

We will be limited here to basic HIS functionalities, handled by most open-source HIS, and with the physical doctor-patient encounter episode.

A. Phase 1. SCOP Applied to the HIS Sub-domain

SCOP input:

- *One quiz and two meetings:* the following general information was elicitated:
- Business goals:
- Facilitate information management to doctors and provide aggregate value to patients with on-line IT via Web, on stand-alone or mobile devices.
- General requirements for Patients:
  - Personal data can be modified by authorized doctors or by patient.
  - A Patient can be registered into the system by a doctor: he cannot enter the system without receiving and answering the invitation generated by the doctor.
  - Images are registered by image laboratories, or other image centers (to be developed): - Data persistency: a package is offered for the period that a patient data has to be kept; - patient is notified for payment.
  - Doctor associates data to patient during the first appointment, allowing him to consult/modify his personal data.
- General requirements for Doctors:
  - He can register to the system personally or have been invited by a patient when he associates him to his data. If he will not answer, he will not be registered by a patient invitation.
  - If no Internet connection is available, manual data transcription (24x7x365) should be managed
- *Business Rules (BR):* use of the proprietary AWS (Amazon Web Services) provider to have *Infrastructure as a Service (IaaS)* cloud configuration, use of *PostgreSQL Database Management System (DBMS)*, use of *Java*[13] language & related platforms

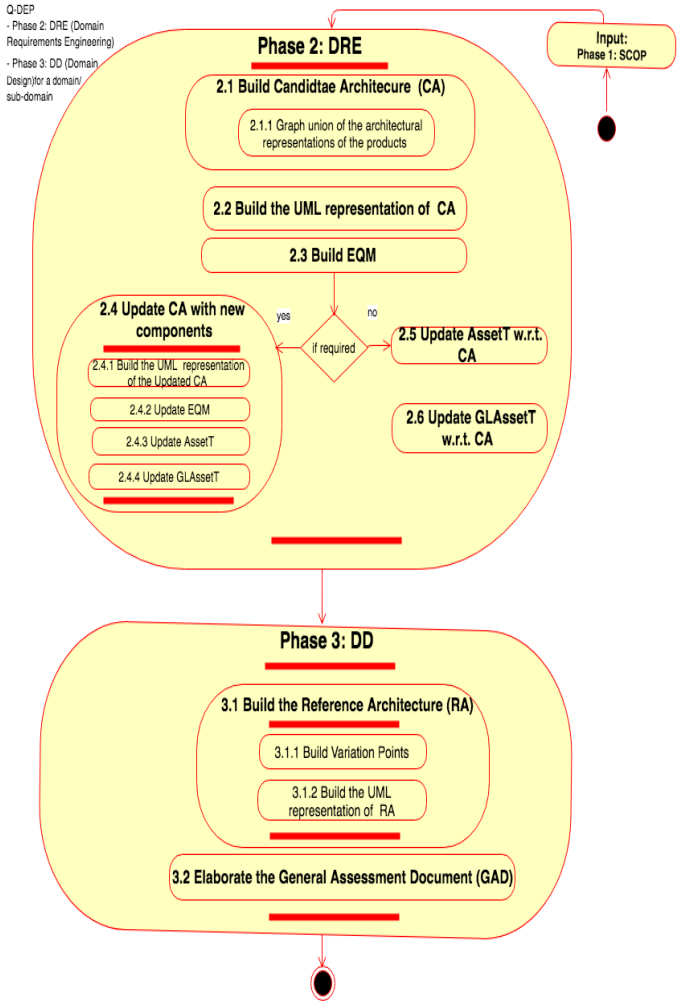

Figure 2: Q-DEP – Phase 2 DRE & Phase 3 DD – UML Activity Diagram

1. Build Product Portfolio (ProdPort) - Activity 1.1:

- Products identified were: *Web Application (WebApp) – for* patient attention services, general medical practice, administration; displayed via a browser and executed through AWS; *Native Application (NatApp)* free download from Apple Store (iOS) and/or Paly Store (Android) and displayed on the client device, offering the same services as WebApp, also through AWS. The UI of

[12] Unified Modeling Language

[13] Sun Microsystems

both products interacts with the AWS cloud services for data process, retrieval and storage; two main components are distinguished: *FrontEnd (Presentation Layer)* and *BackEnd (Process and Data Layers)*; Table I shows main NFR and main support IT mechanisms used, for each product in each layer; a similar product available on the market was *Viewmed online*[14].

- Future products or extensions that could be developed: - assure EHR interoperability with standard formats, like HL7; - Graphical imaging management; - assure EHR availability, since it depends on the cloud; - on-line help to diagnosis; - use of digital electronic payment media.
- Products Portfolio (ProdPort): WebApp, NatApp.

Table I: NFR - FrontEnd – Presentation Layer

FrontEnd NFR	WebApp – IT tools satisfying NFR	NatApp – IT tools satisfying NFR
usability	CSS3[15] for page styles; it depends on the page design	Java and Swift[16] languages for secure messages services
portability	AngularJS[17]-JavaScript[18], HTML5[19]	Swift for iOS and Java for Android SDK [20]
maintainability (modifiability)	AngularJS, with MVC[21] to separate Presentation and Process layers	BackEnd - Ruby-on-Rails[22] provides the RESTful Web service with MVC to send data to NatApp, to separate Presentation and Process Layers
Resources utilization (time efficiency)	Bootstrap[23] FrontEnd framework	Java and Swift languages
Security (authenticity, confidentiality, integrity)	Module developed by the enterprise for access control and role policy	Module developed by the enterprise for access control and role policy

Table II: NFR - BackEnd – Process Layer

BackEnd NFR	WebApp – IT mechanisms satisfying NFR	NatApp – IT mechanisms satisfying NFR
security (authenticity, confidentiality, integrity)	HTTP/HTTPS[24]; system access control, roles policies to access EHR are in a separate module developed by the enterprise	same
portability	Ruby-on-Rails: open source platform and Ruby language; it can work without a specific dada base	same
reliability (robustness)	Ruby-on-Rails	same
reliability (availability-persistency)	PostgreSQL DBMS for data persistency via AWS; the system availability depends on the AWS connection	same
modifiability	Ruby-on-Rails	same
interoperability	AWS cloud services	same

[14] https://www.viewmedonline.com
[15] https://developer.mozilla.org/es/docs/Web/CSS/CSS3
[16] https://www.swift.com
[17] https://angularjs.org
[18] https://www.javascript.com
[19] http://www.w3.org/TR/2014/REC-html5-20141028
[20] https://stuff.mit.edu/afs/sipb/project/android/docs/sdk/index.html
[21] Model, View Controller (GOF pattern), https://wikipedia.org/wiki/Model–view–controller
[22] https://rubyonrails.org
[23] https://getbootstrap.com
[24] Internet Communication Protocols

2. Agile Domain Analysis (ADA) – Activity 1.2:

- *The architectural style(s) used were:* event-based/layers, with a client-server model for distribution and communication.
- *Priority of FR and NFR:* - Priority NFR: security (1), interoperability (1), availability (1), efficiency (time and resources) (2), usability (for the FrontEnd design) (2), maintainability (modifiability) (3), correctness-precision (3), where $1 \leqslant \text{priority} \leqslant 3$; - Priority FR: Appointments and turns control, HER Management (1), Support for Doctoral Practice, for access roles management.
- *Quality Model (DQM) by [ISO 11], considering FR&NFR (Tables I and II) BR&IT, quality properties from architectural styles:* security, portability, maintainability (modifiability, reuse), usability, efficiency (time, resources, scalability), reliability (availability, maturity-robustness), functional suitability (correctness-precision).
- *Similarity Analysis between products' components:* It is shown in Table III, for each product; "R" represent the relation or connector between two components; common components are outlined in grey; the name of the components are taken from Tables I and II; components are labeled sequentially according to the layer and the sub-domain:

Table III: CCT Table showing Components and Connectors

FrontEnd - Presentation Layer		**BackEnd - Process and Data Layers**
WebApp	**NatApp**	**WebApp & NatApp**
a1. FrontEnd	a1	**b1. AWS – BackEnd**
a2. WebPortal	-	**b34. Ruby-on-Rails**
-	a13. UI	**b35. Patient**
a4. In Browser	-	b36. Register Patient
a5.In AngularJS	-	b37. Appointments and turns
a6. In HTML5	-	**b38. EHR Management**
a7. In CSS3	-	**b39. Medical Practice**
a8. In Bootstrap	-	b40. Register Doctor
-	- a14. Java-Android	b41. Examinations
-	- a15. Swift-iOS	b42. Recipes and Medicines
		b43. Medical Reports
		b44. Administration
		b45. Billing
		b46. Laboratory
		b47. Insurance
		b48. Security
		b49. Lab Info
		c1. DBMS
		c2. PostgreSQL
Connectors		**Connectors**
a1Ra2	-	b1Rb34
-	a1Ra13	b34Rb35, b35Rb38, b34Rb39,
-	a13Ra14	b34Rb44, b34Rb48, b34Rc1
-	a13Ra15	b35Rb36, b35Rb37, b35Rb38,
a2Ra4	-	b35Rb39, b35Rb44, b35Rb48

a2Ra5	-	b38Rb39, b38Rb48
a2Ra6	-	b39Rb40, b39Rb41, b39Rb42,
a2Ra7	-	b39Rb43, b39Rb44, b39Rb48,
a2Ra8	-	b39Rc1,
a2Rb1	-	b44Rb45, b44Rb46, b44Rb47,
-	a13Rb34	b44Rb48, b46Rb49
		c1Rc2

- *Architecture of the main products developed:* it is built using the identified components and connectors for each product from CCT Table (see Table III, Figures 3 and 4).
- Notice that in the FrontEnd, excepting the main component *a1. FrontEnd*, there are no common components; user-interfaces are different because one is a Web page for WebApp and the other is a classic stand-alone UI for NatApp; all components in the BackEnd are common.
- For each product developed by the enterprise, the traceability between the components of the architecture is established in the TraT Table (see Table IV).
- Notice also that a separate table should have been built for each product, but in this case only the FrontEnd was different and both tables were integrated to abridge the presentation.

3. Assets Identification - Activity 1.3:

   Main common and variant assets for the sub-domain are determined building the AssetT, see Table V;

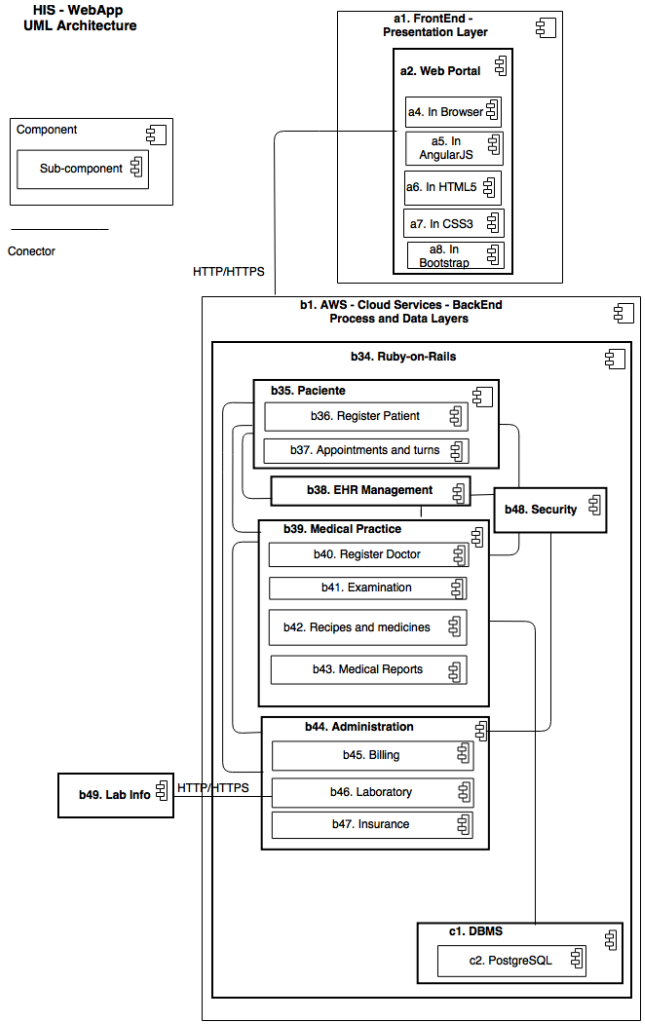

Figure 3: Architecture of WebApp Product Expressed in UML 2.0

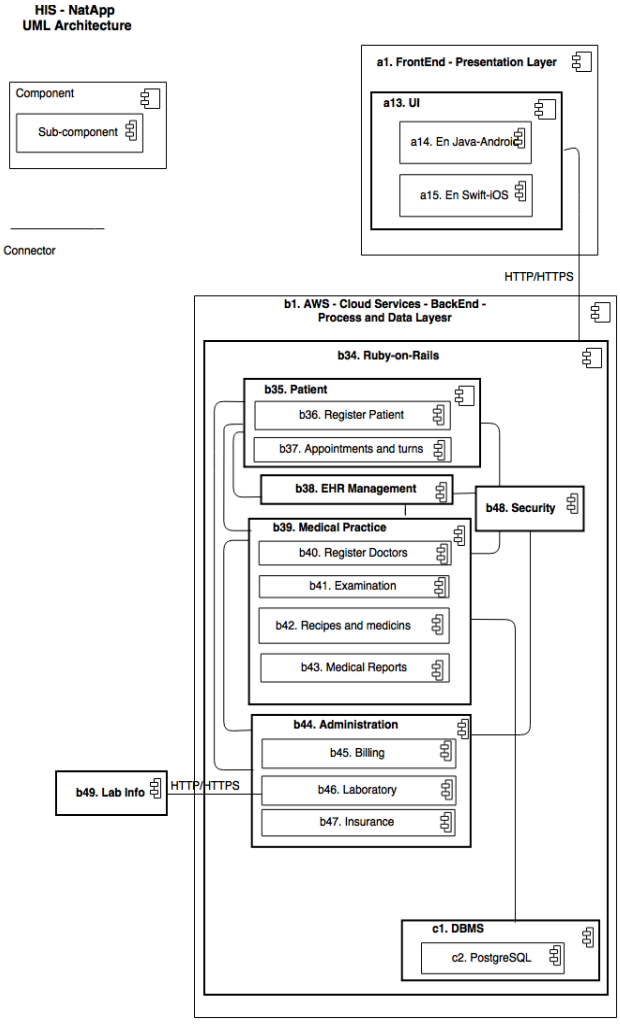

Figure 4: Architecture of NatApp Product Expressed in UML 2.0

Table IV: TraT Table showing Required Quality Properties and IT Components Providing them

WebApp and NatApp Components	Required Quality Property	Provided IT Tool to satisfy Quality Property	Description/Comments
a1. FrontEnd	- availability	- a4. In Browser	- depending on the browser availability
- a2, Web Portal	- usability	- a6. In CSS3	- cascade style for HTML documents
	- modifiability	- a5. In AngularJS	- open framework, JavaScript-MVC
	- efficiency	- a7. In Bootstrap	- open framework for development
	- portability	- a8. In HTML5	- markup language version 5
- a13. UI	- portability	- a14. In Java-Android	- display on different OS with Java and Swift
	- usability	- a15. In Swift-iOS ---	- UI is developed in Java and Swift languages; usability depends on the design, it will not be assured by the architecture
	- modifiability	- b34. Ruby-on-Rails	- Web service *RESTful* of Ruby-on-Rails is provided by the BackEnd
	- availability	- 100% on device	- it is a stand-alone UI; downloaded form App Store (iOS) or Paly Store (Android); the execution of functionalities depend on the availability of the cloud services
	- efficiency	- b34. Ruby-on-Rails	- open-source developing platform
	- security	- b34. Ruby-on-Rails	- module developed by the enterprise for access control and role policy, in the BackEnd
b1. AWS - BackEnd	- portability - interoperability - modifiability	- Cloud services provider	- IaaS; AWS is compliant with the quality of cloud services
	- security	- HTTP/HTTPS	- network communication protocols
	- reliability (availability-persistency, robustness)		
- b34. Ruby-on-Rails	- reliability (availability-persistency, robustness) - portability-modifiability	- Development Platform	- open-source development platform, includes components and sub-components
- b35. Patient	- security (authenticity, confidentiality, integrity)	- b48. Security	- Module developed by the enterprise for access control and role policies
	- availability-persistency	- c2. PostgreSQL	- Microsoft relational DBMS; on-line availability depends on Internet connection since it runs on the cloud; b16. Ruby-on-Rails holds mechanisms for portability to Java objects
- b36. Register Patient	- same as b34	- b34	
- b37. Appointments and turns	- same as b34 + correction-precision	- b34	- handled by b34. It was not specified as a specific component or module
- b38.HER Management	- security (authenticity, confidentiality, integrity)	- b48. Security	- Module developed by the enterprise for access control and role policies
	- availability-persistency	- PostgreSQL	- Microsoft relational DBMS; on-line availability depends on Internet connection since it runs on the cloud; b16. Ruby-on-Rails holds mechanisms for portability to Java objects
	- efficiency (resources-scalability)		
	- interoperability	--- absent --	- to be developed – integration of IT tools for HL7
- b39. Medical Practice	- availability-persistency	- same as b38	
	- security (authenticity, confidentiality, integrity)	- b48. Security	- Module developed in Java by the enterprise for access control and role policies
- b40. Register Doctor	- same as b39	- same as b39	- b40, b41, b42, b43 are sub-components of b39
- b41. Examination	- same as b39	- same as b39	
- b42. Recipes and medicines	- same as b39	- same as b39	
- b43. Medical Reports	- same as b39	- same as b39	

- b44. Administration	- security (authenticity, confidentiality, integrity)	- b48. Security	- Module developed in Java by the enterprise for access control and role policies - handled by b34. It was not specified as a specific component or module
	- correctness-precision	- b34	- handled by b34. It was not specified as a specific component or module
	- efficiency (response time)	- b34	- handled by b34. It was not specified as a specific component or module
- b45. Billing	- interoperability	- Adobe Acrobat	- pdf files
- b46. Laboratory	- the same as b44	- the same as b44	- b45, b46, b47 are sub-components of b44
- b47. Insurance	- the same as b44	- the same as b44	
	- the same as b44	- the same as b44	
- b48. Security	- security (authenticity, confidentiality, integrity)	- Module	- developed in Java by the enterprise for access control and role politicies
- b49. Info Lab	- availability-persistency	- external module to AWS	- it contains laboratory information; used by b46
c1. DBMS	- all DBMS quality properties hold	- Database Management System	- Microsoft relational DBMS; b16. Ruby-on-Rails holds mechanism for portability to Java objects; quality properties are provided by DB mechanisms.
- c2. PostgreSQL	- the same as c1	- same as c1	

Table V: AssetT Table showing main Assets used by Products WebApp and NatApp

Layer	WebApp and NatApp Components	Comments
FrontEnd - Presentation Layer	- a1. FrontEnd - a2. Web Portal - a4. In Browser - a6. In CSS3 - a5. In AngularJS - a7. In Bootstrap - a8. In HTML5 - a13. UI - a14. In Java-Android - a15. In Swift-iOS	- No common assets
BackEnd- Process Layer	- b1. AWS – cloud services - b34. Ruby-on-Rails - b35. Patient - b36. Register Patient - b37. Appointments and turns - b38. HER Management - b39. Medical Practice - b40. Register Doctor - b41. Examination - b42. Recipes and medicines - b43. Medical Reports - b44. Administration - b45. Billing - b46. Laboratory - b47. Insurance - b48. Security - b49. Info Lab - b80. InteropEngine - b81. Imaging	- All assets are common to WebApp and NatApp - b35, …, b48 are common components developed by the enterprise - New component - New component - Internet communication protocols, variants for WebApp and NatApp
Communication Protocols	- HTTP/HTTPS	- Common for layers' interface
BackEnd - Data Layer	- c1. DBMS - c2. PostgreSQL	- Common to WebApp and NatApp; portability to Java objects is responsibility of b16. Ruby-on-Rails

SCOP output:

ProdPort (WebApp, NatAPP), FR (Table I), *NFR* (Table I), *CCT Table* (Table III), *TraT Table* (Table IV), *AssetT Table* (Table V), *WebApp architecture* (Figure 3), *NatApp architecture* (Figure 4).

B. Phase 2. Domain Requirements Engineering (DRE) Applied to the HIS Sub-domain

1. Build Candidate Architecture (CA) – Activity 2.1: Automatic construction of CA by the union of the graphs representing the products' architectures, using

the information provided by SCOP artifacts: ProdPort, CCT and TraT tables.

2. Build UML representation of CA –Activity 2.2:
It is shown in Figure 5. variants are a2 and a13 on the FrontEnd and the connectors to the BackEnd; all other components are common.

3. Build table EQM – Activity 2.3:
Integrate the TraT tables for each product with provided/required quality properties and possible constraints, see Table VI. EQM shows a different CA view, documenting possible scenarios of IT components satisfying quality requirements. In this case the BackEnd is similar for both products and the EQM table is very similar to the TraT table; only the FrontEnd will be shown in Figure 6, as an example of the EQM Table.

4. Update CA with new components – Activity 2.4:
it was not necessary in this case because there were no new components.

5. Update AssetT – Activity 2.5:
with respect to CA which has been automatically built; the new components for interoperability and imaging will be added, they are placed directly in Table V (AssetT Table) to abridge the presentation.

6. Update GLAssetT Table – Activity 2.6:
GLAsset = AssetT in this case, since only one sub-domain is considered.

Table VI: EQM Table

CA Components	Quality Property required by Component	IT Components satisfying Quality Properties	Description/Comments/Constraints
a1. FrontEnd			**Process and Data Layers**
- a2. Web Portal (variant)	- availability	- a4. In Browser	- depending on the browser availability
	- usability	- a6. In CSS3	- cascade style for HTML documents
	- modifiability	- a5. In AngularJS	- open framework, JavaScript-MVC
	- efficiency	- a7. In Bootstrap	- open framework for development
	- portability	- a8. In HTML5	- markup language version 5
	- security	- b34. Ruby-on-Rails	- accessed at logging the system
		- a14. In Java-Android	- display on different OS with Java or Swift
- a13. UI (variant)	- portability	- a15. In Swift-iOS	depending on the device OS
		---	- UI is developed in Java and Swift languages; usability depends on the design, it will not be assured by the architecture
	- usability		
		- b34. Ruby-on-Rails	- Web service *RESTful* (MVC) of Ruby-on-Rails is provided by the BackEnd
	- modifiability		
		- 100% on device	- it is a stand-alone UI; downloaded form App Store (iOS) or Paly Store (Android);
	- availability		- the execution of functionalities depend on the availability of the cloud services
		- b34. Ruby-on-Rails	- open-source developing platform
	- efficiency		
			- a2 and UI, including their sub-components, cannot be present together in a product configuration
	- security	- b34. Ruby-on-Rails	- Accessed at login

C. Phase 3. Domain Design (DD)

1. Build the Reference Architecture (RA) – Activity 3.1:

- *Build Variation Points* by grouping variant components performing similar tasks: there are three variation points for the FrontEnd: - <<*a16. Portal*>>, which includes different portals that can be built considering also the sub-variation point <<*a40. PortDevPlatforms*>> where choices of platforms could be made according to IT changes; - <<*a17. UInterface*>> that can include different stand-alone UI designs and also different IT choices; - <<*a18. UInterfaceConnector*>>, because *a2* and *a13* connect differently *to b1. BackEnd*: *a2* using *b4. AngularJs* who provides the separation of Presentation and Process Layers to get modifiability; *a13* uses instead *b34. Ruby-on-Rails* with the RESTful service for MVC to get also modifiability. It is clear that new IT mechanisms will continue to appear on the market, similar to those used by the enterprise and other cloud services providers besides AWS, and other developing platforms besides *b34. Ruby-on-Rails* could be adopted, hence two new variability points are included in the BackEnd, <<*b82. Cloud Services*>> and <<*b83. BEDevPlatforms*>>, because can be also changed, providing more genericity to this HIS RA.
- *Build the RA UML diagram representing the architecture*: it can be seen in Figure 6. New components that have been stated in Future products are included, *b80. InteropEngine* for HL7 standard and *b81. Imaging* to handle graphic imaging.

2. Elaborate the GAD Document – Activity 3.2:

The GAD document contains basically a summary of the artifacts obtained, limitations and recommendations. Among the limitation:

- The architecture documentation (components and connectors) [16] was almost absent for the enterprise, being reduced to just an incomplete deployment diagram; however it was found that now with the UML 2.0 architectural diagrams provided, deployment diagrams could more be easily designed, knowing explicitly all the components that were using.
- No standards were used.

Among the recommendations:

- Maintain updated all the tables and diagrams in case of changes, since they are the reusable assets, including the RA diagrams or documentation.
- Incorporate as soon as possible a standard format for EHR.
- The GLAssetT Table should be implemented as a "real" Core Asset Repository of the artifacts produced.

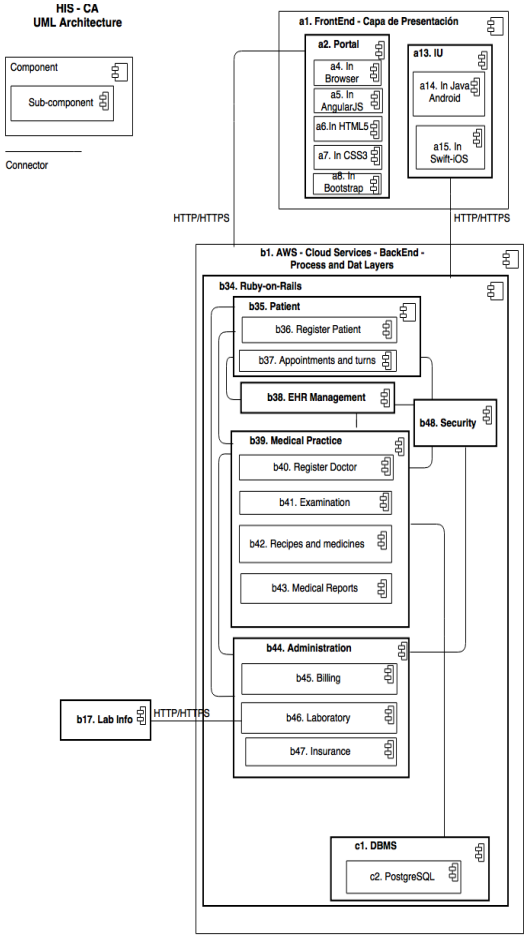

Figure 5: HIS CA Expressed in UML 2.0: Union of WebApp and NatApp Graphs' Architectural Representations

V. CONCLUSION

The Q-DEP extractive bottom-up quality-driven domain engineering process for SPL has been presented. Our aim was to relate the experience in applying a "light weighted" SPL engineering approach that can be effectively used in industrial practice. Q-DEP has been entirely developed for this experience, adapting the QuaDRA process [24] to a real industrial context. Both processes are centered on the first PLScope phase of DE to reduce the efforts in the subsequent phases, but Q-DEP does not use business processes specifications to identify main functionalities and constraints; using a "semi-agile" practice more adapted to the MC domain, it profits from interviewing stakeholders and performing meetings to arrive to an agreement on the existing products architecture. It was applied to the MC domain, facing an increasing market demand and where no software engineering good practices are yet employed, due to the rapid time-to market delivery requirement. The experience with a small-medium sized software enterprise was satisfying, three different sub-domains of MC were studied successively (financial transactions, healthcare information systems and entertainment contents), and the components of the products

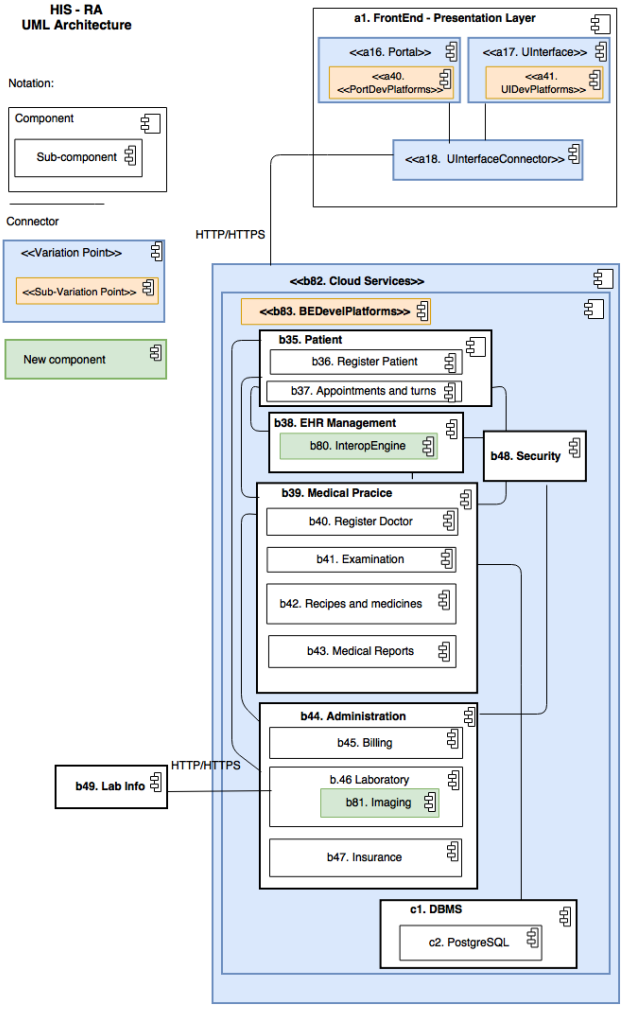

Figure 6: HIS-RA Expressed in UML 2.0

developed were identified by direct interaction with project leaders. We did not pretend to construct a complete SPL, but just to start a migration by building a first glance of the enterprise Core Asset Repository and the RA of the sub-

domains considered. In this paper we applied Q-DEP to the Healthcare Information Systems sub-domain, showing and explaining the construction of all the artifacts produced. In Q-DEP quality is considered from the start. Software development in MC is based on reusing lots of IT support tools, and this fact has been useful showing implicit satisfaction of the quality goals required by MC applications; developers use IT tools transparently, generally unaware of the quality aspects involved, but in this way the quality of their applications is assured "by construction" to a certain extent. As IT evolves quickly, there is hope that support tools will also improve and implicitly continue to guarantee the quality of mobile software. Another aspect that has to be signaled is that several design choices, such as the selection of the variation points, are mostly based on the architect expertise and on his vision of the domain evolution, but this is difficult avoid in architectural design. On the other hand, a limitation of Q-DEP is that automatic support tools are under construction and they are necessary to face complexity of intensive systems; besides, the integration of the different RAs of the three sub-domains into a global RA for the enterprise is still an ongoing work, a new sub-process has to be defined and integrated to Q-DEP.

Among the perspectives, we are working on an ontological approach to represent the RA [30] and the Asset Repository, to derive consistency rules for the AE cycle. A comparison of the ISO/IEC 26550 versions of 2013 and 2015 is also planned.

ACKNOWLEDGMENT

We wish to thank María Dolores Nardi, Aquilino Pinto, Carlos Alfonzo and Mary Gutiérrez, to make this work possible with fruitful comments and observations.

REFERENCES

[1] J. Bosch, *Design and Use of Software Architectures – Adopting and Evolving a Product-line Approach*, Addison-Wesley, 2000.

[2] P. Clements and L. Northrop, *SPL: Practices and Patterns*, 3rd edition. Readings, MA, Addison Wesley, 2001.

[3] K. Pohl, G. Böckle, and F. van der Linden, *SPL Engineering - Foundations, Principles, and Techniques*. Springer IXXVI, 2005.

[4] I. Reinhartz-Berger, A. Sturm, T. Clark, S. Cohen, and J. Bettin. *Domain Engineering. Product Lines, Languages and Conceptual Models*, Springer, 2013.

[5] ISO/IEC NP 26550, *Software and Systems Engineering – Reference Model for Software and Systems PL*, ISO/IEC JTC1/SC7 WG4, 2013.

[6] E. Berard, *Essays in OO Software Engineering*, Prentice Hall, N.Y., 1992.

[7] P. Abrahamsson and P. Keynote: *Mobile Software Development - The Business Opportunity of Today*, in proceedings of the International Conference on Software Development, pp. 20-23. Reykjavik, Island, 2005.

[8] A. Wasserman, *Software Engineering Issues for Mobile Application Developmen*t, FoSER'10, FSE/SDP on Future of Software Engineering Research, pp. 397-400, 2010.

[9] ISO/IEC 25010, *Systems and Software Engineering - Systems and Software Quality Requirements and Evaluation (SQuaRE) - System and Software Quality Models*, ISO/IEC JTC1/SC7/WG6, 2011.

[10] F. Losavio, O. Ordaz, and V. Esteller, *Refactoring-Based Design of Reference Architecture*, RACCIS, vol. 5, no. 1, pp. 32-48.

[11] World Wide Web Consortium, *Mobile Web Application Best Practices*, W3C Working Draft, http://www.w3.org/TR/mwabp, 2010.

[12] Agile Alliance, *Agile Software Development Manifesto.* Retrieved from Manifesto for Agile Software Develop. http://agilemanifesto.org, 2001.

[13] B. Boehm and R. Turner, *Balancing Agility and Discipline: A Guide for the Perplexed*. Addison-Wesley, 2003.

[14] A. Spataru, *Agile Development Methods for Mobile App*, MSc. These, University of Edinburg, Edinburg, UK, 2010.

[15] B. Kaleel and S. Harishankar, *Applying Agile Methodology in Mobile Software Engineering: Android Application Development and its Challenges*, Reyrson University, Reyrson, Canada, CSTR. Paper 4, http://digitalcommons.ryerson.ca/compsci_techrpts/4, 2013.

[16] M. Shaw and D. Garlan, *Software Architecture. Perspectives of an Emerging Discipline*, Prentice-Hall, 1996.

[17] J. Herrera, F. Losavio, O. Ordaz, and J. Bøegh, *Product Lines Scoping Using ISO/IEC 26550 Reference Model Considering Software Quality*, ASSIT2016, Bangkok, Thailand, pp. 194-200, indexed by CPCI-SSH, ISBN 9791605953885, http://www.ichss2016.com, July 2016.

[18] I. Mansanet, J. Fons, I. Torres, and V. Pelechano, *GeMMINi: Prototipo de IU Sobre Multiples Dispositivos. Una Estrategia basadas en LPS y MDD*, FAZ, 2011.

[19] K. Czarnecki, C. Hwan, P. Kim, and K. Trygve, *Feature Models are Views on Ontologies*, SPIC, 2006.

[20] T. Gruber, *Toward Principles for the Design of Ontologies Used for Knowledge Sharing*. Technical Report KSL 93-04, KS Laboratory, Stanford University, CA, USA, 1993.

[21] L. Bass, M. Klein, and F. Bachmann, *Quality Attribute Design Primitives and the ADD Method*, SEI White Paper, http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=29604, 2001.

[22] H. Koziolek, R. Weiss, and J. Doppelhamer, *Evolving Industrial Software Architectures into a Software Product Line: A Case Study*. LNCS 5581, pp. 177-193, 2009.

[23] F. Losavio, O. Ordaz, N. Levy, and A. Baiotto, *Refactoring Process for Software Product Lines Design*, (JDLP, pp. 47-58, Lille, France, http://jldp.org/2012/images/jldp2012-actes.pdf, Novembre 2012.

[24] J. C. Herrera, F. Losavio, and O. Ordaz, *QuaDRA: Quality-oriented Design of Reference Architecture for Software Product Lines Based on ISO/IEC 26550*, RACCIS Vol. 6, No.1, pp. 20-38, Enero-junio, ISSN 2248-7441, http://www.fundacioniai.org/raccis, 2016.

[25] L. Chung, B. Nixon, and E. Yu, *Using NFR to Systematically Select Among Alternatives in Architectural Design*, 1st International Workshop on Architectures for Software Systems, pp.31-42, Seattle, USA, 1995.

[26] D. Bjørner, *Software Engineering 3: Domains, Requirements, and Software Design*, Texts in Theoretical Computer Science, Springer-Verlag, Berlin Heidelberg, Germany, 2006.

[27] F. Losavio, O. Ordaz, and I. Santos, *Proceso de Análisis del Dominio Agil de Sistemas Integrados de Salud en un Contexto Venezolano*, ENL@CE, vol. 12, no. 1, pp. 101-134, ISSN 1690-7515, 2015, http://www.produccioncientifica.luz.edu.ve/index.php/enlace/index.

[28] C. Salinesi, R. Mazo, O. Djebbi, and D. Dia, *Constraints: The Core of Product Line Engineering*, RCIS'11, pp 1-10, Gosier, Guadalupe, 2011.

[29] Object Management (OMG), *Unified Modelling Language Superstructure, Version 2.0* (formal/05-07-04), August 2005.

[30] F. Losavio, O. Ordaz, and S. Jean, *Ontological Approach to Derive Product Configurations from a Software Product Line Reference Architecture*, Revista CyT, UP, Argentina, no. 16, pp. 91-127, ISSN 1850-0870, May 2016, http://www.palermo.edu/ingenieria/pdf2016/CyT_16_07.pdf.

[31] L. M. Northrop and P. C. Clements with collaborators F. Bachmann, J. Bergey, G. Chastek, S. Cohen, P. Donohoe, L. Jones, R. Krut, R. Little, J. McGregor, and L. O'Brien, *Framework for Software Product Line Practice*, Version 5.0. SEI, Carneige Mellon University, December 2012.

[32] P. Krutchen, *Architectural Blueprints — The "4+1" View Model of Software Architec*ture, IEEE Software, vol. 12, no. 6, pp. 42-50, November 1999.

Revista Venezolana de Computación
ISSN: 2244-7040
http://www.svc.net.ve/revecom
Vol. 4, No. 1, pp. 22-33, Junio 2017
Fecha recepción: 28/04/2017, Fecha aceptación: 09/08/2017

Hacia un Enfoque para la Generación de Código con MDA

Mercedes Picón[1], Javier Torrealba[2]
mpicon@una.edu.ve, jtorrealba@una.edu.ve

[1] Centro Local Metropolitano-Ingeniería de Sistemas, Universidad Nacional Abierta, Caracas, Venezuela
[2] Coordinación de Ingeniería de Sistemas, Universidad Nacional Abierta, Caracas, Venezuela

Resumen: Se presentan opciones disponibles y posibles riesgos a considerar al seleccionar herramientas y métodos para generar código con MDA. Con el objetivo de complementar la información suministrada por los proveedores de herramientas MDA, se propone un enfoque que guíe la selección de generadores automáticos de código. Para reducir el riesgo que suponen los procesos pesados asociados con frecuencia a MDA, se incluyeron prácticas Ágiles en el enfoque propuesto. Muchas herramientas apoyan la generación de código con base en modelos UML y las herramientas seleccionadas van a influir en todas las fases de los proyectos de desarrollo y en el mantenimiento de sistemas. Sin embargo, las referencias citadas afirman que se trata de una tecnología en evolución y que está por alcanzar su madurez.

Palabras Clave: Generadores de Código; UML; MDA; Selección de Herramientas MDA; Prácticas Agile; Codificación.

Abstract: Available options and potential risks to consider are presented regarding the selection of tools and methods to generate code with MDA. To supplement information provided by MDA tool vendors, an approach is proposed to guide the selection of automatic code generators. To reduce risks frequently related to heavy-weight MDA processes, Agile practices have been included in the proposed approach. Many tools support UML based code generation and the selected tools will impact all project development phases and system's maintenance. However, the cited references claim it's an evolving technology which has yet to reach maturity.

Keywords: Code Generators; UML; MDA; MDA Tool Selection; Agile Practices; Coding.

I. Introducción

Mediante prácticas estándar de programación es posible producir en forma automática código completo y correcto basado en diagramas UML (Unified Modeling Language) o Lenguaje Unificado de Modelado [1]. A partir de diagramas de estado, se ha producido código para componentes de comportamiento de alta complejidad en sistemas de tiempo real y embebidos [1]. Sin embargo, aún mediante herramientas que apoyan la MDA (Model Driven Architecture) o Arquitectura Guiada por Modelos, particularmente aquellas que transforman modelos en código, no logra producirse la totalidad del código para algunos componentes de software [2][3][4].

Los métodos y herramientas de MDA para generar código en forma automatizada están evolucionando hacia su madurez [3][5], pero todavía se carece de metodologías con procesos que puedan ser instanciados para apoyar MDA y en particular, para orientar la selección de generadores de código basados en UML (GC) [4][6]. Con el propósito de diseñar un enfoque que pueda guiar la generación de código mediante GC, se investigaron prácticas de la MDA y su integración con metodologías Ágiles.

A partir de esta introducción, el presente artículo está organizado en las siguientes secciones: en la Sección II se describe brevemente el enfoque MDA; la Sección III presenta prácticas con modelos UML para la generación automática de código; en la Sección IV se describen características de las herramientas MDA para la generación automática de código. Una clasificación de posibles riesgos en los procesos anteriores se ofrece en la Sección V. La Sección VI propone un enfoque para emplear MDA, conjuntamente con prácticas ágiles en la generación automática de código y finalmente, la Sección VII presenta las conclusiones que resultaron de esta investigación.

II. El Enfoque MDA

La MDA plantea la obtención de valor de los modelos y del modelado, como apoyo al manejo de la complejidad e interdependencia de sistemas complejos [7]. En 2001 el OMG presentó la MDA como un enfoque para la MDE (Model Driven Engineering) o Ingeniería Guiada por Modelos. MDE es un enfoque que surgió para incrementar el nivel de abstracción en la especificación de programas y la automatización en el desarrollo de programas [8]. Los estándares de la MDA incluyen los siguientes [7]:

- XMI (XML Metadata Interchange) como mecanismo estándar para intercambio,
- UML como lenguaje para modelado mediante diagramas,
- CWM (Common Warehouse Metamodel) como estándar para almacenes de datos y
- MOF (Meta Object Facility) como mecanismo para analizar modelos en XMI, el cual provee los constructos estándar para modelado e intercambio en la MDA.

UML está conformado por una familia de lenguajes para especificar, visualizar, construir y documentar el diseño y la estructura de sistemas de software [9][10]. UML 2.0 comprende diagramas dentro de las categorías de estructura, de comportamiento y de interacción [9]. En el análisis y diseño bajo el enfoque OO (Orientado a Objetos), se desarrollan diagramas UML hasta obtener un formato utilizable por los programadores [11].

MDA define los siguientes niveles de modelado con UML [7]:

- CIM (Computation Independent Model) o Modelo Independiente de Cómputo, se emplea para representar los modelos de negocio y requerimientos de sistemas [12]
- PIM (Platform Independent Model) o Modelo Independiente de Plataforma, es generado sin detalles sobre las soluciones técnicas y es convertible en uno o más modelos de plataforma específica [13]
- PSM (Platform Specific Model) o Modelo de Plataforma Específica, puede convertirse en el código diseñado para una plataforma específica [13]. Se produce a partir del PIM mediante herramientas MDA [4]. Al cambiar los requerimientos del negocio se modifica el PIM en vez los modelos PSM, los cuales se regeneran a partir del PIM modificado [14].

Al separar los modelos PIM de la información técnica relacionada con plataformas específicas, la MDA incrementa la reusabilidad de PIMs y PSMs [15]. Los modelos PIM y PSM son de interés particular en la generación automática de código basada en UML. No obstante, no hay un método estándar para construir modelos CIM ni para transformarlos en uno o varios PIM en niveles más bajos de abstracción, siendo el PIM el objetivo final de la construcción de modelos que permitan la generación de código [12][14].

El MDD (Model Driven Development) o Desarrollo basado en Modelos es un subconjunto de MDE que se concentra en transformaciones que generan representaciones más concretas a partir de abstracciones [13][16][17]. Plantea desarrollar modelos extensos antes de escribir código fuente. Tanto MDD como MDA pueden categorizarse como sub-secciones de MDE [17].

Un estudio de seis metodologías basadas en MDA concluyó que no ofrecen un proceso de desarrollo que se pueda instanciar [4]. La selección de GC no es apoyada en las metodologías analizadas en ese estudio. Se desarrolló una metodología basada en MDA que se considera posible de instanciar [4]. Sin embargo, en su actividad para selección de herramientas solo recomienda determinar cuáles son los requerimientos para estas herramientas MDA y compararlos con las capacidades de herramientas disponibles en el mercado.

III. Prácticas Basadas en Modelos UML para la Generación Automática de Código

Se describen prácticas basadas tanto en enfoques prescriptivos como en metodologías ágiles. Estas prácticas han sido sugeridas por desarrolladores de métodos y herramientas MDA, por proyectos que emplearon GC y por estudios acerca de métodos y herramientas. Se discuten en esta sección porque son necesarias para la codificación automatizada y para ajustar los modelos durante iteraciones que conduzcan a elevados porcentajes de código correcto.

MDA presenta pocos detalles a seguir para las transformaciones de un nivel de modelado a otro hasta obtener el código [2]. Los equipos que han desarrollado herramientas que transforman modelos a modelos o a código, así como también los proyectos que emplearon GC, proveen prácticas y recomendaciones a considerar en el desarrollo y mantenimiento de sistemas con MDA. Se presentan en primer lugar las pertinentes al modelado UML y en segundo lugar, las relacionadas con los dos tipos de enfoques de MDA.

A. Prácticas de Modelado

El diagrama de clase es el punto inicial para generar la estructura básica del código. Posteriormente, el cuerpo de los métodos se complementa con diagramas de secuencia y diagramas de máquinas de estado [5]. Los diagramas de casos de uso son empleados principalmente en modelos CIM. PIM utiliza todos los diagramas UML excepto los de componente y de despliegue. PSM tiene diagramas de componente y de clase. Algunos proyectos expresan PSM como formatos estándar de código y comentarios significativos, lo que es aceptado por la perspectiva y los principios ágiles [13].

La propuesta de Agile UP describe lo que puede considerarse una ampliación de los diagramas UML asociados a la Arquitectura de las 4+1 vistas, para distintos objetivos de modelado [18][19]. La Tabla I muestra solo los diagramas UML correspondientes a los tipos de modelo identificados [18].

Tabla I: Tipos de Modelado y sus Diagramas UML

Modelado	Diagramas UML utilizados en tipos de Modelado
de Utilización	Casos de Uso
Conceptual de Dominio	Clases
de Procesos	Actividad
Arquitectónico	Componentes, Despliegue, Paquetes
de Objetos Dinámicos	Comunicación, Estructura Compuesta, Resumen de Interacción, Secuencia, Máquina de Estado, Temporización
Estructural detallado	Clases, Objeto

Los modelos proveen un nivel de abstracción que facilita la representación de la información arquitectónica y de diseño, así como la corrección de fallas de arquitectura y de diseño, la eliminación de los aspectos estructurales y funcionales innecesarios y la adaptación de la estructura para lograr funcionalidad adicional y optimización del comportamiento [10]. Sin embargo, un solo modelo es insuficiente para desarrollar un sistema de software completo, se requieren varios modelos en distintos lenguajes de modelado, e incluso código en lenguajes de programación [2].

Se pueden necesitar diagramas adicionales a los presentados la Tabla I [18][19], para evitar que el modelado con UML resulte en una representación incompleta de los componentes y de su código. Por ejemplo, para el modelado de la interfaz usuaria y el de requerimientos suplementarios, se recomiendan diagramas distintos a los de UML [18].

B. Prácticas que Combinan MDA y Metodologías Ágiles

Entre las iniciativas para integrar los enfoques de MDA con las propuestas de modelos ágiles están:

- Agile MDA, basada en la noción de que el código y los modelos ejecutables son operativamente lo mismo y, por lo tanto, los principios ágiles aplican a los modelos [17].
- AMDD (Agile MDD) promueve el uso de herramientas de apoyo simples y diagramas UML para modelar requerimientos con un proceso iterativo e incremental [20].

El enfoque de transformación se utiliza para producir modelos o artefactos de una tecnología específica; el de modelos ejecutables, recomendado para prácticas ágiles [10], se implementa en una plataforma tecnológica que ejecuta el modelo directamente, tratándolo como código fuente interpretable. Ambos enfoques logran la generación de código ejecutable a partir de modelos [7].

En el enfoque Ágil MDA, los modelos deben ser suficientemente completos para que puedan ser ejecutados por sí solos sin que sea necesario distinguir entre modelos a nivel de análisis y de diseño. En vez de transformar modelos, estos se conectan entre sí y se mapean hacia un único modelo combinado que, por último, es traducido al código según un sistema único de arquitectura MDA [13].

Mediante UML ejecutable (xUML), MDA ha logrado mejoras en la calidad del software al habilitar modelos UML para que puedan traducirse en código ejecutable [10]. Un modelo ejecutable UML puede tratarse como un PIM, de acuerdo con los conceptos de MDA [15].

Aunque xUML tiene limitaciones, se ha propuesto combinar este enfoque con prácticas ágiles, particularmente las que generan el código de pruebas automáticamente. Se ha considerado adecuado para sistemas embebidos en dispositivos dentro de los cuales en software no puede fallar [10]. Esto se ha justificado indicando que las prácticas ágiles como son XP y ASD, evitan separar las actividades de diseño de las de implementación y abandonan las actividades de documentación, para favorecer el desarrollo de pruebas rigurosas imprescindibles en sistemas de alta criticidad [10].

Existe una falta de compatibilidad de principios entre XP y MDA; la primera enfatiza aspectos medulares de la tecnología de software sin basarse en diagramas para documentar un sistema. Estudios que examinaron la integración de XP y el modelado con UML, recomiendan el Modelado Extremo como solución [21].

La forma frecuente de combinar MDA con metodologías ágiles es incorporar al enfoque MDA prácticas que incluyen XP (Extreme Programming), Scrum, DSDM (Dynamic Systems Development Method), ASD (Adaptive Software Development) y Crystal [17]. Las tres primeras son las más adoptadas en desarrollos de sistemas basados en modelos [17].

De una revisión realizada por Hansson y Zao, que abarcó 24 artículos de IEEE, 27 de ACM y 27 de Springerlink [17], se presenta a continuación una relación entre las prácticas ágiles y MDA, basada en los trabajos de Zhang y Patels:

- Modelado como codificación: UML como lenguaje de programación visual que es compilado por el GC ha generado C o C++
- Modelado iterativo e incremental: Se modela en UML durante múltiples sprints, repitiendo el mismo conjunto de actividades en cada sprint. Un modelo ejecutable evoluciona en cada sprint con un incremento funcional con respecto al modelo del sprint anterior. Los modelos se mantienen ejecutables tanto para la simulación como para pruebas en la plataforma de ejecución
- Modelado continuo: Corresponde a la práctica ágil de la integración continua que unifica los diseños en UML frecuentemente, con la generación automática de código para varias simulaciones en un sprint; se produce también en forma automática el código para pruebas dos o más veces en un sprint
- Codificación: generación de código C/C++ completamente ejecutable basado en diagramas de máquinas de estado orientadas a transiciones.

Con los mismos diagramas UML se puede generar el código fuente y el de pruebas. Esto aumenta la disponibilidad de las pruebas al inicio de las actividades de codificación. Con el enfoque "probar primero" los casos de prueba pueden basarse en diagramas de secuencia. La generación automática de código de programas y del código de pruebas, contribuye a identificar errores en los modelos [10].

Es necesario definir métodos que proporcionen pautas a seguir y técnicas a utilizar para desarrollar software con MDA [20]. Puede integrarse MDA con metodologías robustas como Rational Unified Process (RUP) o livianas como las propuestas ágiles, de forma que MDA provea los estándares y mecanismos para concebir una arquitectura alrededor de modelos y su proceso de transformación, y que las metodologías suministren lineamientos para la gestión del proyecto [20].

El uso de prácticas ágiles en el desarrollo de sistemas de tiempo real tendría que estar a cargo de expertos tanto en el dominio del sistema como en los métodos ágiles. Sin embargo, se considera que el rigor y robustez que aportan los modelos permite que MDA combinado con Agile pueda ser la base para desarrollar componentes de alta criticidad [17].

Las prácticas descritas anteriormente deben considerarse al fijar los criterios de selección de GC.

IV. Características de las Herramientas MDA para Generar Código con Base en Modelos UML

Se describen a continuación las características que deberían considerarse al seleccionar generadores de código basados en modelos UML (GC). Un GC es una herramienta que transforma un modelo conformado por diagramas UML en código fuente en lenguajes de alto nivel [1]. El modelo puede

ser un PSM que describa la estructura, el comportamiento o la arquitectura de los sistemas a desarrollar [22].

Estas herramientas incluyen ArgoUML, Rational Rhapsody, StarUML, Altova UModel, y Acceleo [23][24][25][26][27], entre otras. Sus características, extraídas principalmente de los manuales de estas herramientas, sirven para establecer los criterios de selección.

Entre las características de funcionalidad (idoneidad funcional) se encuentran las presentadas a continuación:

- Generación de código a nivel de producción en lenguajes: C, C++, Java, C#, Visual Basic, Perl, Python, PHP, entre otros.
- Generación de código para pruebas unitarias, de integración y de sistema [10].
- Diagramas UML procesados para generar código.
- Versión de UML: 2.4, 2.1, 2, 1.3.
- "Ida y vuelta" o "round-trip" para crear y ajustar diagramas a partir del código.
- Enfoque de generación de código: estructural, de comportamiento o de intercambio [1][28].
- Generación de código para definir y manipular datos en DBMS específicos.
- Integración con verificadores de modelos como SPIN [1].
- Capacidades de DSL (Domain Specific Language) o lenguajes de dominio específico.
- Transformaciones modelo a modelo CIM-PIM-PSM.
- Capacidades para diferenciación, comparación y refactorización de modelos, revisión de consistencia de modelos (Ej. SPIN), gestión de versiones e integración de modelos y co-evolución de modelos [2].
- Conformidad con estándares MDA.
- Uso especializado o de propósito general.

Entre las características de compatibilidad (coexistencia e interoperabilidad), están las siguientes relacionadas con los ambientes de desarrollo y operaciones:

- Plataforma Operativa: Linux, Mac OS X, MS Windows.
- Formato de archivo para intercambio de información: OCL (Object Constraint Language), XMI (XML Metadata Interchange), .uml (de StarUML).
- DBMS.
- Integración con ambientes y herramientas de desarrollo: Visual SourceSafe, CVS, MS Word, Eclipse, Visual Studio.NET.
- Integración con herramientas para transformación de modelos y con herramientas de pruebas de software, entre otros procesos de desarrollo.

Otras características de los GC incluyen:

- Proveedor: código abierto o comercial.
- Continuidad de la herramienta y proveedor
- Estabilidad de la versión.
- Servicios de Soporte técnico del proveedor.

Se detallan a continuación algunas de las características anteriores con recomendaciones para evaluar las herramientas:

- Diagramas UML procesados para generar código. Algunos GC procesan hasta 14 diagramas UML 2.4 [25]; otros no procesan diagramas de secuencia, de objeto, paquetes, temporización, ni de entorno de interacción; varios GC solo apoyan aspectos fundamentales de las máquinas de estado según la especificación UML 2.0 [15].
- Existen GC que producen código fuente genérico para aplicaciones que van a operar en una variedad de plataformas [29].
- Existen GC que pueden mapear entre distintas fuentes y destinos [25].
- "Ida y vuelta" o "round-trip" para crear y ajustar diagramas a partir del código [25]. Combinar la ingeniería hacia adelante y la ingeniería en reverso para obtener diagramas UML a partir del código y así documentar componentes de los sistemas, generar los diagramas UML mediante ingeniería en reverso y obtener PSM reutilizables.
- Los DSLs incluyen Domain Specific Modeling Languages (DSML) o lenguajes para modelado de dominio específico, los cuales se construyen con lenguajes meta-modelo de propósito general como MOF [30].
- Transformaciones Modelo a Modelo. Hay herramientas que facilitan la creación de un modelo PIM y su transformación a un modelo PSM. También transforman el modelo PSM en un código totalmente ejecutable en varios lenguajes de programación [2]. Por el contrario, hay transformaciones de PIM a PSM realizadas con lenguajes o plantillas propietarias, sin conformidad con OMG MDA. Estas transformaciones pueden además producir código incompleto [31].
- Para facilitar estas transformaciones algunos GC mejoran o corrigen los diagramas [1][28].
- Uso especializado o de propósito general. Los GC disponibles en la actualidad se han empleado en el dominio automotriz, en sistemas embebidos y de tiempo real, con requerimientos exigentes de confiabilidad. Sin embargo, UML no cubre las necesidades de modelado de los distintos dominios, por lo que pueden necesitarse GC especializados u otros métodos para generar código [10].
- Conformidad con estándares MDA. Incluye conformidad con subconjuntos de UML como fUML para la generación de código mediante modelos ejecutables [8].
- Plataforma Operativa y Sistema manejador de BD. Aunque el código generado tenga dependencias con ciertos sistemas de bases de datos y sistemas operativos, varias herramientas generan código ejecutable sobre una variedad de plataformas [29], incluyendo plataformas móviles. Existen GC que permiten generar código ajustado a un DBMS [25][26].
- Integración con ambientes y herramientas de desarrollo: Los métodos y herramientas GC deben integrarse con las plataformas de desarrollo y operacional. Los GC pueden ser componentes de Integrated Development Environments (IDE) o ambientes integrados de desarrollo que tienen conformidad con MDA.

Entre las funciones que ofrecen los GC de carácter experimental o académico están las siguientes:

- Generación de un PSM que representa N-Capas para Web con enlaces de trazabilidad entre los componentes de meta-modelos PIM y PSM y reglas de transformación con el lenguaje Atlas Transformation Language (ATL) [16].
- Transformación de máquinas de estado en UML 2.0 a C#; considerando todos los conceptos de estos diagramas y apoyando el desarrollo eficiente de aplicaciones multi-hilos, bien documentadas y con capacidad de mantenimiento [15].
- Generación del código C++ completo de un reloj digital y un juego de Tetris con base en diagramas de máquinas de estado [1].
- Producción de código Java a partir de diagramas de secuencia creados con un plug-in de Eclipse como IDE, para modelado con UML [32].

Un estudio realizado en el 2015 concluye que ninguna herramienta por sí sola puede apoyar las distintas actividades de MDA [2]. Entre estas actividades que complementan la generación de código destacan [2]:

- la interacción entre el modelo y el código así como entre modelos;
- transformaciones modelo a código, modelo a modelo, y código a modelo;
- funcionalidad para la cual se cuenta con menos apoyo como la integración de modelos, mapeo entre modelos, y fusión de modelos.

Las actividades anteriores serán de utilidad en las iteraciones de la fase de construcción para ajustar los modelos de forma que se eleve el porcentaje de código completo y correcto generado.

V. Riesgos en la Generación Automática de Código con Modelos UML

Los siguientes riesgos se pueden presentar en los procesos de MDA relacionados con la generación automatizada de código. Estos se clasifican aquí en dos categorías, según estén más relacionados con los métodos o con las herramientas:

A. Riesgos Asociados a MDA o UML

1) Elaboración de los Modelos: Riesgos relacionados con los diagramas que conforman los modelos y con su elaboración.

Los diagramas pueden estar elaborados en exceso o en forma insuficiente. Al elaborar los diagramas UML durante el modelado pueden omitirse atributos indispensables para generar código y los diagramas tendrían que completarse para ser procesados por los GC. Esto se debe en parte al énfasis de los diagramas en la comunicación usuario-analista.

Los diagramas creados en las iteraciones de los flujos de trabajo de requisitos, análisis y diseño, intentan comunicar modelos con diferente nivel de detalle, entre usuarios y desarrolladores [11]. Estos diagramas pueden estar insuficientemente elaborados para generar código; pueden omitir elementos indispensables para codificar, ej. el recuadro que indica bucles o repeticiones en un diagrama de secuencia.

Al trasformar CIMs a PIMs o PIMs a PSMs, la pérdida de información puede impedir la generación de código. También es posible que deban elaborarse más los modelos para llegar al PIM con los diagramas y nivel de elaboración adecuados para generar código completo. Un modelo incompleto puede incluir diagramas UML sin suficiente elaboración para permitir la programación manual o automática.

2) MDA: Este enfoque presenta los siguientes riesgos posibles en relación con el uso de GC.

La modalidad con UML ejecutable propone que el sistema se especifique en su totalidad dentro del PIM. Este enfoque propuesto por Shlaer-Mellor afirma que a partir del PIM tanto el PSM como el código son generados en un 100% [33]. Sin embargo, la existencia de dos modalidades para implementar MDA, plantea la posibilidad de una elección inapropiada [33].

En la otra modalidad hay enfoques de transformaciones de modelos que implementan en forma inadecuada las actuales especificaciones de UML. Un ejemplo es la automatización insuficiente para adaptar los diagramas de secuencia al modificar los de estado y para ajustar los de objeto al cambiar los diagramas de clase correspondientes. Esto influye en la transformación automática de modelos a código correcto [10] y favorece la selección de la primera modalidad de MDA.

MDA cambia el énfasis del desarrollo desde los programas hacia los modelos [29]. Su integración con las prácticas ágiles debe considerar que estas no promueven el modelado, ni la construcción de dos modelos, uno de análisis y otro de diseño. Aunque se afirma que combinar MDA y prácticas ágiles reduce las limitaciones de ambos enfoques [17], los desarrolladores tendrían contar con experiencia previa en esta integración para reducir el riesgo del proyecto.

MDA plantea que con PIM se crea el PSM en forma automática, entonces se trata de tener un PIM completo teniendo en cuenta que MDA no prescribe ningún proceso de desarrollo específico para desarrollarlo. Cada proyecto de desarrollo con MDA debe definir su propio proceso de desarrollo o seleccionar un proceso dentro de un conjunto muy pequeño de metodologías MDA disponibles [34].

3) UML: Riesgos relacionados con UML como lenguaje de modelado.

El diseño de lenguajes de modelado posee una base teórica reducida [1]. Carece también de un mapeo completo entre diagramas UML y código [35]. Estas limitaciones pueden permitir inconsistencias al transformar los diagramas a código como en el caso de no generar asociaciones que pueden ser necesarias entre clases [28]. La limitación anterior puede dificultar el establecimiento de relaciones precisas entre diagramas UML y tipos de componente como la interfaz usuaria, el manejo de base de datos y la lógica empresarial.

Acerca de la relación entre la lógica de comportamiento y la capa de lógica empresarial, es notoria la falta de información que permita mapear capas de aplicación y diagramas [3].

UML está lejos de cubrir las distintas necesidades de proyectos que difieren en su dominio de aplicación, tamaño, necesidades de confiabilidad y presión por tiempos de entrega [10]. Para complementar la generación de código basada en UML, puede ser necesario utilizar DSL y/o pseudocódigo [29]. Los DSL

permiten un énfasis en decisiones de diseño pertinentes al dominio y permiten el empleo de los conceptos y abstracciones más adecuados.

La generación automática de código basada en UML puede disminuir la calidad de los siguientes atributos [29]:

- Capacidad de mantenimiento: a) discontinuidad e incompatibilidad de las nuevas versiones de los GC, b) estructura del código que dificulta su lectura y comprensión.
- Certificación: a) la inspección y análisis de código se dificultan debido a su gran extensión y a la baja comprensibilidad de su estructura. Se han desarrollado GC que mejoraron un diagrama de estado para reducir las líneas de código generadas a partir del mismo [1][28]; b) el código puede depender en tiempo de ejecución, de librerías sin código fuente ni documentación, que puedan introducir vulnerabilidades en los componentes.
- Portabilidad: los estándares MDA del OMG ofrecen portabilidad para nuevo hardware e infraestructuras mediante modelos de software tan complejos que pueden reducir los beneficios del GC.

B. Riesgos o Limitaciones Asociados a la Generación de Código con Herramientas MDA

Se presentan riesgos documentados en la literatura especializada, relacionados con el uso de GC.

El código generado para un clasificador en UML dado, puede ser solo una plantilla de código con estructuras como definiciones de clase, atributos con su tipo de dato, sin código alguno, a llenar por el usuario [3][27]. Puede ser necesaria la inclusión manual de código para la conectividad de base de datos con la interfaz de usuario, cajas de texto, comandos para botones, botones de radio, casillas de chequeo, entre otros, así como para el manejo de tablas de base de datos [3].

Entre los riesgos o limitaciones asociados a herramientas específicas se encuentran:

- Uso de formatos propietarios no estándar para almacenar los diagramas.
- En 2005 existían herramientas con dificultades si un diagrama de estado estaba sobrecargado, lo cual se simplificaba con pseudocódigo logrando así generar menos líneas de código [36]. Esta dificultad puede evitarse convirtiendo el diagrama a un formato XMI u OCL como uno de los pasos iniciales del GC para producir el código.
- Diagramas no procesados [22].

Los riesgos descritos se han tomado en cuenta para proponer el enfoque que se presenta a continuación.

VI. Enfoque Propuesto para Seleccionar GC

Solo algunas herramientas basadas en UML se implementan con una metodología particular [9]. Las actividades presentadas en la Tabla II se proponen como una aproximación hacia un método para la selección de GC, dentro del marco de la MDA, de las fases del Proceso Unificado (UP) y de las prácticas ágiles aplicables en la construcción mediante GC [37].

Estas actividades tienen por objetivo suplementar la insuficiencia de MDA como enfoque para desarrollar software, según se describió en la Sección II. Se proponen tanto para modelos PIM estables como para modelos que evolucionan con varias iteraciones durante la construcción de componentes.

Tabla II: Actividades Propuestas para Seleccionar GC

Modelos disponibles	**Fases del Proceso Unificado**			
	Iniciación	**Elaboración**	**Cons-trucción**	**Tran-sición**
CIM/PIM	**1.** Definir la estrategia para la generación automática de código. **2.** Seleccionar los GC.			
PIM		**3.** Evaluar los GC seleccionados. **4.** Definir el enfoque de integración de los GC.		
PSM		**5.** Generar el código de componentes y pruebas unitarias.		

El gerente del proyecto, los analistas, diseñadores y probadores pueden participar en el enfoque propuesto, pero los responsables principales son una pareja conformada por un modelador y un programador que inician la actividad 1, imitando la organización de equipos ágiles. Pueden conformarse parejas adicionales al avanzar el proceso, por cada paquete o capa a desarrollar.

A. Actividad 1. Definir la Estrategia para la Generación Automática de Código

Esta actividad se presenta en la Figura 1. Su objetivo es asegurar que el proceso para generar código pueda integrarse con las metodologías y herramientas de desarrollo.

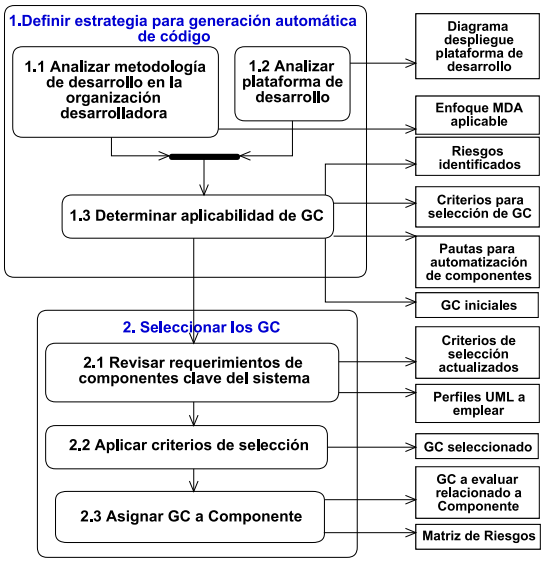

Figura 1: Enfoque Propuesto – Actividades 1 y 2

1) Sub-actividad 1.1. Analizar la Metodología de Desarrollo de la Organización Desarrolladora: Analizar las metodologias y sus métodos, procedimientos, técnicas y herramientas en uso, en relación con el proceso de generación automática de código.

2) Sub-actividad 1.2 Analizar la Plataforma de Desarrollo: Analizar la documentación del software instalado en la plataforma de desarrollo. Se crea un diagrama de despliegue destacando los componentes que forman parte de los procesos de modelado y construcción. Al igual que la sub-actividad anterior, va a facilitar la integración de los enfoques, técnicas y herramientas de desarrollo.

3) Sub-actividad 1.3 Determinar la Aplicabilidad de la Generación Automática de Código Basada en Modelos UML: Determinar cómo se integra la generación automática de código al ambiente de desarrollo, identificando riesgos potenciales, criterios para selección de GCs compatibles, pautas de automatización y una lista inicial con los GC a seleccionar.

Se ejecuta una vez analizados los métodos y herramientas de todas las fases del desarrollo vigentes en la organización, incluyendo las estrategias de adquisición [29], considerando que MDA es apropiado para proyectos grandes ejecutados de una forma muy estructurada y que seleccionar herramientas eficientes y adaptar los procesos de desarrollo existentes a un nuevo proceso de desarrollo ágil y centrado en modelos, puede resultar una tarea muy retadora [10].

Esta sub-actividad incluye determinar si se cuenta con un PIM para construir un prototipo de viabilidad, de bajo costo y desechable, para cada GC a evaluar.

Los productos principales creados en esta actividad son:

- Diagrama similar al de la Figura 2, que represente la integración existente entre las herramientas de modelado, IDE, DBMS y los GC. Varios modeladores UML incluyen un GC [9][23][27].

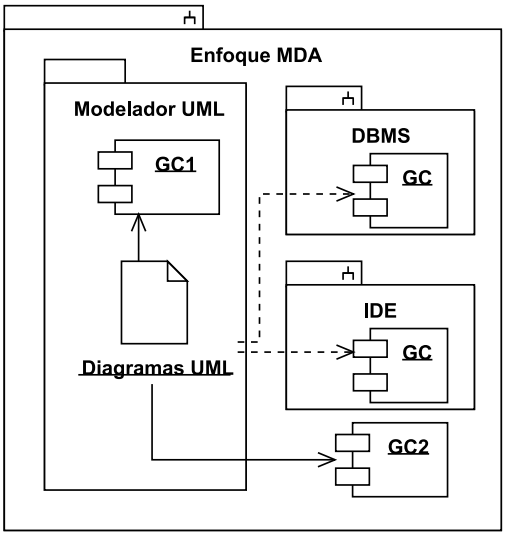

Figura 2: Componentes del Ambiente de Desarrollo

- Enfoque de MDA aplicable: UML ejecutable o transformación. Los modelos UML-ejecutables se consideran el enfoque Agile de MDA que puede reducir la fase de modelado. Deben ser integrados, a diferencia de transformados, para construir la funcionalidad del sistema [13]. La aplicación de UML ejecutable con prácticas Ágiles, debería aumentar la eficiencia del desarrollo del proyecto [10][21].
- Lista de riesgos a mitigar. Posteriormente se refinan y actualizan al crear una matriz de riesgos. El seguimiento de los riesgos relacionados con los componentes en cada sprint o iteración será apoyada por la gestión de riesgos del proyecto global.
- Entre los criterios de selección, se determina si se eligen solo los GC o además las herramientas MDA, para transformaciones Modelo a Modelo.
- Pautas para automatización. Estas incluyen: a) el tipo de componente a generar mediante los GC, indicando su paquete o capa; b) métodos para codificar: mediante GC conformes con estándares MDA, con GCs sin certificaciones de OMG MDA, con DSL, pseudocódigo o en forma manual c) no utilizar GC cuando exista el método probado para la operación de una clase, debido al escaso beneficio de generar el diagrama de actividad para un algoritmo de ordenamiento y codificarlo mediante un GC.

B. Actividad 2. Seleccionar los GC

Esta actividad se presenta en la Figura 1. Su objetivo es seleccionar los GC candidatos a evaluar en la actividad 3. La selección de los GC impacta todo el ciclo de vida de desarrollo de sistemas y, siendo numerosas las herramientas MDA que pueden incluir la funcionalidad de los GC [14][27], se determinan las candidatas a una posterior evaluación.

1) Sub-actividad 2.1 Revisar Requerimientos de Componentes Clave del Sistema a Construir: Mediante el análisis de los requerimientos de los componentes clave del sistema, modelados en los diagramas UML, se determina si los modelos están lo "suficientemente completos" para generar código que permita evaluar los GC.

Para cada componente clave se identifica su capa de aplicación y si su código es de comportamiento, estructura o interacción. Esta información debe estar disponible para la selección de los GC y permitirá actualizar y ampliar los criterios formulados anteriormente, como resultado de analizar los modelos existentes.

Si un componente clave no tiene disponible un PIM, se puede modelar el dominio del sistema con un diagrama de clases y adicionalmente modelar el comportamiento externo del componente con un diagrama de secuencia [12].

2) Sub-actividad 2.2. Aplicar los Criterios de Selección: Asignar pesos a los criterios de selección y aplicarlos a los GC.

Para aplicar los criterios es recomendable disponer del GC instalado en una plataforma que permita generar el código de

componentes clave con base en los diagramas y determinar el nivel de cumplimiento de cada criterio de selección por el GC.

La Tabla III muestra los criterios de selección de GC, entre otros, a aplicar en esta sub-actividad. Estos criterios se proponen de acuerdo a las características descritas en la Sección IV.

Tabla III: Características para la Selección y Evaluación de Generadores Automáticos de Código basados en UML (GC)

Características para seleccionar y evaluar los GC	S	E
Funcionalidad		
- Código generado a nivel de producción: C, C++, Java, C#, Visual Basic, Perl, Python, PHP.	√	
- Código generado para pruebas.		
- Diagramas UML procesados y no procesados o con fallas.	√	√
- Versión de UML: 2.4, 2.1, 2, 1.3.	√	
- "Round-trip" para crear y ajustar diagramas a partir del código.		
- Enfoque de generación de código: estructural, comportamiento, intercambio.	√	√
- Generar código para definir y manipular datos en DBMS específicos.		
- Integración con verificadores de modelos como SPIN.		
- Capacidades de lenguajes de dominio específico (DSL, siglas en inglés).		
- Transformación CIM, PIM y PSM.	√	√
- Ejecución de modelos.	√	√
Compatibilidad con los ambientes de desarrollo y operaciones		
- Plataforma Operativa: Linux, Mac OS X, MS Windows, Android.	√	
- DBMS.		
- Formato de archivo para intercambio de información: OCL, XMI, .uml (de StarUML).		
- Integración con ambientes/herramientas: CVS, Eclipse, Visual Studio.NET., entre otros.		
- Integración con herramientas para transformar modelos CIM-PIM-PSM		√
Otras características		
- Proveedor: código abierto o comercial.	√	
- Uso especializado o de propósito general.	√	
- Continuidad de la herramienta y estabilidad de la versión.	√	

Se recomienda tener en cuenta lo siguiente al aplicar los criterios de selección:

- Aplicar en primer lugar, los criterios a los GC del modelador UML y del IDE,
- En segundo lugar se aplican a los GC de proveedores reconocidos identificados en la actividad 1 y actualizados en la sub-actividad 2.1.
- Los GC que apoyan el desarrollo de sistemas de tres capas - presentación, lógica empresarial y persistencia - pueden ser adecuados para los requerimientos exigentes de sistemas de tiempo real y embebidos, en cuanto a capacidad de mantenimiento, eficiencia, seguridad y tolerancia a fallas, entre otros requerimientos no funcionales.
- La compatibilidad con: a) las herramientas que almacenan los modelos de análisis y con otros métodos y herramientas adoptados por la organización o equipo de desarrollo [29], b) la plataforma de desarrollo y de operaciones incluyendo el IDE y el DBMS y c) estándares OMG MDA incluyendo especificaciones de UML [27].

Para esta selección preliminar, se utilizan los criterios tildados en la columna titulada "S", los cuales tendrían que adaptarse y tener pesos asignados según las actividades anteriores. Esta sub-actividad puede requerir entrevistas a proveedores, aplicación de cuestionarios, análisis de experiencias de otros proyectos, revisión de especificaciones en manuales y el uso del GC.

Los criterios tildados bajo la columna "E" probablemente requieran más recursos y tiempo para evaluar, incluso pueden necesitar modelos PIM estables para componentes críticos.

La Figura 3 representa los criterios considerados más determinantes para seleccionar los GC, debido a la estrecha relación que estos tienen con el código generado. Se consideran criterios relacionados con la idoneidad funcional de acuerdo con los conceptos, prácticas y funcionalidades de GC, descritos en las secciones anteriores. Se recomienda aplicar los criterios de selección de los GC en cuanto a los tres tipos de código y capas identificadas para los componentes en la sub-actividad 2.1, considerando el nivel del modelo disponible.

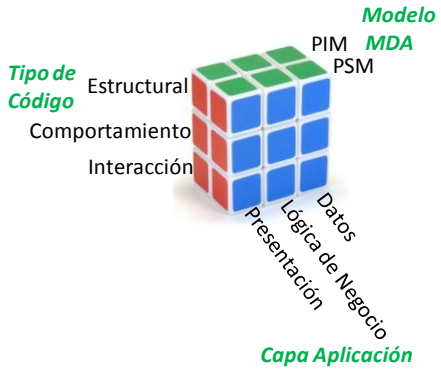

Figura 3: Criterios de Idoneidad Funcional para Seleccionar Generadores de Código

Si existen componentes a construir con código de comportamiento, debido a que este código se genera a partir de los diagramas de máquinas de estado, de componentes, de secuencia y de actividad [27], los GC se seleccionan si procesan estos diagramas.

2) Sub-actividad 2.3. Asignar GC a Componentes: Esta asignación se apoya en las especificaciones del GC.

UJECTOR es un GC que especifica la siguiente relación entre diagramas UML 2.0 y código fuente [28]: de clase, para generar el esqueleto del código Java; de secuencia, para los métodos y finalmente, de actividad y de secuencia, para la manipulación de objetos e interacción con el usuario.

En esta actividad se determina si el GC será asignado a uno o más componentes para ser evaluado. Se identifican riesgos y limitaciones como es el caso de diagramas no procesados por el GC. Puede ejecutarse durante la creación del PIM. Requiere identificar todos los diagramas que representan cada capa en el PIM y haber determinado si el GC los procesa utilizando para ello, las capacidades del GC para transformar automáticamente el PIM en uno o más PSM.

Los productos generales resultantes de la actividad 2 son:

- Los criterios para seleccionar los GC basados en sus características y capacidades.
- Perfiles de UML a emplear como es el caso de: perfil ITU-T Z.109 de UML para sistemas distribuidos; MARTE para sistemas embebidos de tiempo real; jUML y fUML para xUML ejecutable [2].
- La asociación Componente-GC para los componentes clave. Esta se representa en una matriz la cual se refinará en las actividades siguientes, revisando de nuevo los aspectos anteriores al contar con el modelo PIM completo.
- Matriz de riesgos basada en la lista preliminar obtenida en la actividad 1, actualizada de acuerdo a los mencionados en la Sección V para los GC.

C. *Actividad 3. Evaluar los GC Seleccionados*

Puede ejecutarse en forma paralela a la anterior; su objetivo es determinar si los GC seleccionados generan código correcto y completo para cada capa y tipo de código (comportamiento, estructura e interacción) del sistema a construir. Se indican en la Figura 4 sus sub-actividades y productos:

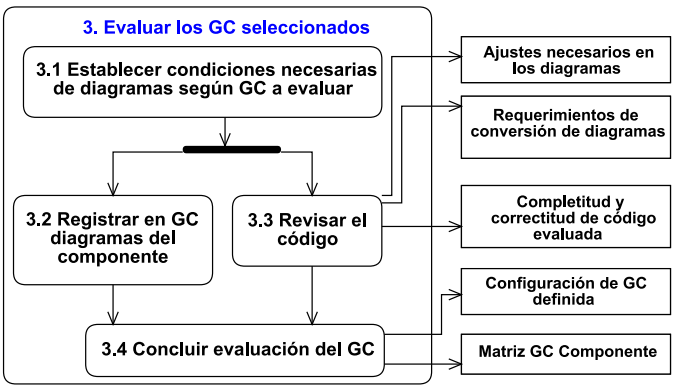

Figura 4: Enfoque Propuesto – Actividad 3

1) Sub-actividad 3.1. Establecer las Condiciones Necesarias en los Diagramas según los GC a Evaluar: Esta subactividad requiere que el PIM esté disponible. Los diagramas UML recomendados a continuación, podrán ser utilidad para analizar el PIM y luego completarlo:

- Los relacionados a los tipos de código de acuerdo con el OMG UML [9],
- Los relacionados a las 4+1 vistas de la organización de un sistema de software [19] y
- Los indicados en la Tabla I [18].

Luego se determinan las condiciones que deben reunir los diagramas para un GC particular. El UP recomienda evitar detalles innecesarios con demasiada anticipación a la construcción [11]. Esto puede resultar en un modelo con diagramas que habrán de ser ajustados a las condiciones que imponen los GC.

Puede ser necesario completar los modelos con el apoyo de la ingeniería en reverso, transformando el código que cumpla los requerimientos del sistema a desarrollar. El modelo resultante puede compararse con el PIM incompleto que se obtuvo en la ingeniería hacia adelante, a fin de completar sus diagramas.

2) Sub-actividad 3.2. Registrar en el GC los Diagramas del Modelo que Representa el Componente a Codificar: Unos GC generan código para paquetes de diagramas además de generar código para un diagrama específico.

La documentación de los GC puede describir la relación entre los componentes frecuentemente usados, como son los componentes de las capas de aplicaciones Web, y los diagramas indispensables como parte de un modelo.

3) Sub-actividad 3.3. Revisar el Código Generado: Se ejecuta mediante inspecciones que determinan el porcentaje generado y si cubre los requerimientos representados en el modelo.

Las inspecciones incluyen: a) el nivel de adecuación para los tipos de código y para estándares de calidad según los requerimientos; b) la consistencia entre el diagrama y el código y c) el número y complejidad de los ajustes requeridos en los diagramas.

Las sub-actividades 3.2 y 3.3 son iteraciones modelado-codificación-prueba para evaluar tanto el código generado como la facilidad de ejecución de las iteraciones.

4) Sub-actividad 3.4. Concluir la Evaluación: Decidir e informar si el GC va a generar el tipo de componente para el cual fue evaluado.

Se requiere la siguiente información actualizada: a) el componente a desarrollar, la capa del sistema a la cual pertenece, el tipo de enfoque de su código y todos los diagramas UML que lo representan y b) los ajustes a realizar en los diagramas UML según los requerimientos que impone el GC.

Los productos principales generados en esta actividad son:

- los ajustes en los diagramas indispensables para generar código, considerando cómo el GC implementa los conceptos de UML
- requerimientos de conversión de los diagramas para su procesamiento por un GC
- el nivel de completitud y correctitud del código generado; será necesario contar con el código de las pruebas para determinar si el código de los componentes clave funciona correctamente
- la configuración de cada GC con el cual se van a crear los componentes y sus pruebas
- matriz GC-Componente actualizada indicando el porcentaje de código, diagramas sin soporte del GC, aspectos por resolver y los requerimientos de conversión.

D. *Actividad 4. Definir la Integración de los GC*

Las sub-actividades y productos de esta actividad se presentan en la Figura 5. El objetivo es definir cómo se integran los métodos y herramientas de forma que esté disponible el ambiente de desarrollo necesario para generar el código.

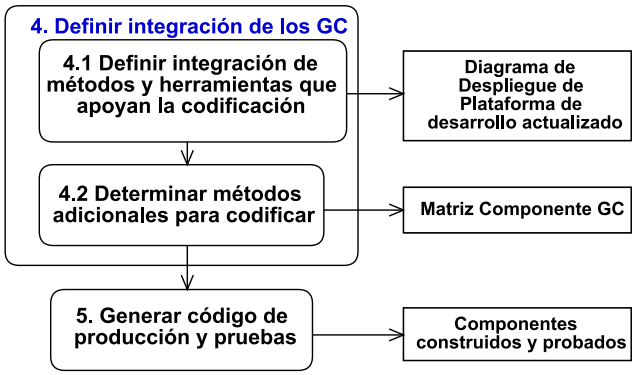

Figura 5: Enfoque Propuesto – Actividades 4 y 5

1) Sub-actividad 4.1. Definir la Integración de Métodos y Herramientas que Apoyan la Codificación: Definir cómo se integran los GC entre sí y con las demás herramientas de las plataformas de desarrollo y de producción.

Esta sub-actividad atiende a la necesidad de utilizar varias herramientas para generar el código completo de diversos componentes. Puede completarse durante la ejecución de las actividades 1 y 3, para sistemas sencillos de baja complejidad y componentes poco diversos.

2) Sub-actividad 4.2. Determinar el Mapeo Componente-Otro Método de Codificación: Actualizar la matriz Componente-GC, tomando en cuenta la metodología del proveedor y los procesos de conversión necesarios.

En la actividad anterior, al concluir la evaluación del GC, se identificaron los componentes sin apoyo de GC, los cuales se crean mediante métodos a documentar en esta sub-actividad.

Los productos principales generados en esta actividad son:

- Diagrama similar al de la Figura 2 que represente la nueva integración entre herramientas de modelado, IDE, DBMS y GC.
- Matriz Componente-GC actualizada con el método de codificación definitivo: GC, DSL, pseudocódigo, programación manual, entre otros. La opción para generar código se registra en la matriz creada en la actividad 2, y actualizada en la actividad 3.
- Opcionalmente pueden generarse la actualización de la configuración de los GC y la creación de nuevos procedimientos de integración.

E. Actividad 5. Generar el Código Fuente de los Componentes y de las Pruebas

Actividad opcional presentada en la Figura 5; su objetivo es generar el código de los componentes y de las pruebas mediante las herramientas identificadas en la actividad 3 e integradas en la actividad anterior.

Aunque la evaluación del GC para un componente clave específico se completó en la actividad 3, cuando se trata de componentes críticos para la misión de la organización o si la configuración del GC pudiera variar después de la actividad 4, puede ser necesario generar la totalidad del código del componente como parte de su evaluación definitiva.

El producto que resulta de esta actividad es el código generado para los componentes clave, incluyendo el código de pruebas, de forma que pueda afirmarse que el componente se codificó y se probó. Al generar el código se deben tomar en cuenta las siguientes recomendaciones:

- Se genera el código mediante el uso del GC sin necesidad de completar un gran diseño inicial. La generación de código se puede iniciar con modelos en un estado "suficientemente bueno" los cuales serán completados durante el uso de los GC. Esto tiene como ventaja adicional, que permite la detección oportuna de fallas en requerimientos suplementarios o no funcionales como el desempeño [34].
- Solo en el proceso de generación se agregan elementos dependientes de la plataforma. Cuando la tecnología cambia solo se requiere actualizar el GC, pero los modelos que definen la aplicación pueden ser directamente reutilizados [10].
- Los diagramas se ajustan hasta obtener código completo y correcto. Para asegurar la calidad de la programación, se modifica el diagrama antes que el código.
- Las funciones de ingeniería en reverso y "round-trip" permiten verificar la consistencia entre el código fuente generado y los diagramas, y también, mantener los diagramas actualizados en forma automática al actualizar el código.
- Modelando en pareja por un sprint y con rotación de parejas para detectar y resolver problemas de modelado en forma instantánea, se puede ejecutar esta actividad aplicando las prácticas recopiladas que combinan metodologías ágiles y MDA [17], descritas en la Sección III.

Una vez presentadas las cinco actividades propuestas, cabe agregar que al igual que XP y ASD evitan distinguir entre las actividades de diseño y las de implementación [10], las actividades anteriores se ejecutan de acuerdo con el avance del desarrollo de los modelos, y no con referencia a flujos de trabajo propios del Proceso Unificado.

Las actividades descritas se pueden adaptar con enfoques de prototipos de diseño que evolucionen hacia prototipos de producción o implementación.

El enfoque propuesto puede evolucionar mediante su prueba en un proyecto piloto. Se revisaron aspectos mejorables identificados al contrastar el enfoque con buenas prácticas recomendadas para el desarrollo de metodologías [38]. De esa evaluación se identificaron mejoras referidas a la completitud de los elementos involucrados y a la documentación. En trabajos futuros se implementarán estas mejoras, entre otras. Se considera que el enfoque puede guiar selecciones de GC en proyectos de baja complejidad para el desarrollo de sistemas sin criticidad alta para la misión de la organización del usuario.

VII. CONCLUSIONES

Hoy en día, las herramientas MDA disponibles presentan niveles distintos de conformidad con estándares del OMG; adicionalmente, varían en cuanto a las funciones que ofrecen, lo cual hace necesaria una selección rigurosa y exhaustiva que pueda seguir un proceso factible de instanciar.

En la actualidad, las herramientas MDA están más cerca de proveer altos niveles de automatización y reusabilidad aunque sin suministrar la totalidad del código, particularmente el código de comportamiento. Se requieren varios modelos en distintos lenguajes de modelado, e incluso código en lenguajes de programación para desarrollar un sistema de software completo [2].

La generación automática de código basada en modelos cambia el diseño tanto o más que la implementación, y coloca el énfasis más en los distintos niveles de modelos que en la programación. Aunque los GC pueden seleccionarse principalmente por sus capacidades para generar código con base en diagramas UML, un criterio que recientemente tiende a ser más aceptado es la capacidad de transformaciones modelo a modelo y modelo a código.

Los riesgos señalados podrán superarse a medida que evolucione la MDA y sus herramientas. Existen desarrollos de sistemas críticos para la misión que consideran la generación automatizada de código basada en UML como más confiable y segura, sobre todo si cumple con los estándares de calidad del OMG MDA. Al evolucionar y mejorar los GC, la codificación manual pudiera considerarse riesgosa e injustificada y la verificación para certificar una aplicación podrá requerir el uso de los GC compatibles con dichos estándares.

Por el contrario, pudiera repetirse lo ocurrido con los CASE hace más de una década, en cuanto a la falta de apoyo en las transformaciones necesarias para derivar PSM a partir de modelos PIM. Varias herramientas realizan transformaciones directamente de un modelo PIM a código fuente mediante un lenguaje de plantilla en forma inapropiada. Sería de esperar que la iniciativa MOF 2.0/QVT del OMG progrese hacia la estandarización de las transformaciones de modelos [31].

Desde hace décadas, los usuarios disponen de sistemas con componentes configurables en su totalidad y de DBMS que proporcionaban funcionalidad completa para sistemas de procesamiento de transacciones. Dentro de entornos donde se dispone de esas soluciones, los proveedores de GC se adaptan para responder a requerimientos fuertes y propios de sistemas de tiempo real, embebidos, complejos y de alta criticidad. Esa adaptación se apoya en la MDA, la cual continua su evolución hacia lo que se espera sean niveles de madurez y de aceptación por la comunidad de desarrolladores.

Las prácticas ágiles pueden reducir el potencial de los riesgos causados por procesos pesados de desarrollo que se consideran parte del enfoque MDA [17]. Esto debe tomarse en cuenta dentro del proceso de gestión de riesgos del proyecto.

Como trabajo futuro se prevé comparar el código producido por distintas herramientas GC y ajustar el enfoque diseñado, según los resultados de esa comparación y según las prácticas sugeridas por los proveedores de esas herramientas.

REFERENCIAS

[1] M. Rincón, J. Aguilar, F. Hidrobo, *Generación Automática de Código a Partir de Máquinas de Estado Finito*, Universidad de los Andes, Mérida, Venezuela. Computación y Sistemas, vol. 14, no. 4, pp. 405-421, ISSN 1405-5546, 2011.

[2] M. Karanam, *MDA Tool Support for Model Driven Software Evolution: A Survey*, International Journal of Computer Science Engineering (IJCSE), ISSN: 2319-7323, vol. 4, no. 01, January 2015.

[3] S. D. Rathod, *Automatic Code Generation with Business Logic by Capturing Attributes from User Interface via XML*, International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT), Channai, India, pp. 1480-1484, 2016.

[4] M. Asadi, M. Ravakhah, R. Ramsin, *An MDA-based System Development Lifecycle*, Second Asia International Conference on Modelling & Simulation, Asia International Conference on Modelling & Simulation, Kuala Lumpur, Malaysia, DOI:10.1109/AMS.2008.19, 2008.

[5] C. M. Zapata, J. J. Chaverra, *Una Mirada Conceptual a la Generación Automática de Código*, Revista EIA, Escuela de Ingeniería de Antioquia, Medellín, Colombia. ISSN 1794-1237, no. 13, pp. 143-154, Julio 2010.

[6] A. Noureen, A. Amjad, F. Azam, *Model Driven Architecture - Issues, Challenges and Future Directions*, Journal of Software, vol. 11, no. 9, September 2016.

[7] OMG, Object Management Group, *Model Driven Architecture (MDA), MDA Guide rev. 2.0*, OMG Document ormsc/2014-06-01, 2014.

[8] S. Motogna, I. Lazar, B. Parv, I. Czibula, *An Agile MDA Approach for Service-Oriented Components*, Electronic Notes in Theoretical Computer Science 253, 95–110, 2009.

[9] UML.org. http://www.uml.org/what-is-uml.htm.

[10] B. Rumpe, *Agile Modeling with the UML*, Radical Innovations of Software and Systems Engineering in the Future. 9th International Workshop, RISSEF 2002. Venice, Italy, October 2002.

[11] S.R. Schach, *Análisis y Diseño OO con UML y el Proceso Unificado*, 4ta edición, McGraw Hill, 2005.

[12] B. Bousetta, O. El Beggar, T. Gadi, *A Methodology for CIM Modelling and its Transformation to PIM*, Journal of Information Engineering and Applications. vol. 3, no. 2, 2013.

[13] A. El-Sheikh, A. Omran, *Suggested Framework for Agile MDA and Agile Methodologies*, The Research Bulletin of Jordan ACM, ISSN: 2078–7952, Vol. II (III) 2011.

[14] T. Calic, S. Dascalu, D. Egbert, *Tools for MDA Software Development: Evaluation Criteria and Set of Desirable Features*, IEEE DOI 10.1109/ITNG.2008.241, © 2008.

[15] R. Pilitowski and A. Dereziñska, *Code Generation and Execution Framework for UML 2.0 Classes and State Machines.* In: Sobh T. (eds) Innovations and Advanced Techniques in Computer and Information Sciences and Engineering. Springer, Dordrecht, 2007.

[16] M. Rahmouni and S. Mbarki, *Model-Driven Generation: From Models to MVC2 Web Applications*, International Journal of Software Engineering and its Applications, vol. 8, no. 7, pp. 73-94, 2014.

[17] S. Hansson and Y. Zhao. *How MAD Are We? Empirical Evidence for Model-Driven Agile Development*, Bachelor of Science Thesis, Software Engineering and Management. University of Gothenburg, Chalmers University of Technology, Department of Computer Science and Engineering, Göteborg, Sweden, June 2014.

[18] S. W. Ambler, http://www.agiledata.org/essays/enterpriseArchitecture Techniques.html. Ambysoft Inc. Copyright 2002-2013.

[19] FCG Software Services (FCGSS), *Applying 4+1 View Architecture with UML 2*, White Paper, Copyright ©2007.

[20] J. B. Quintero, R. Anaya, *MDA y el Papel de los Modelos en el Proceso de Desarrollo de Software*, Revista de la Escuela de Ingeniería de Antioquía. no. 8, Julio/Diciembre, 2007.

[21] B. Rumpe, *Executable Modeling with UML - A Vision or a Nightmare?* - Issues & Trends of Information Technology Management in Contemporary Associations, Seattle. Idea Group Publishing, Hershey, London, pp. 697-701, 2002.

[22] A. Mateen, A. Tabassum, *A Framework for Model Driven Transformation Engineering towards Software Architecture and Performance*, International Journal of Computer Applications, vol. 143, no. 8, June 2016.

[23] A. Ramirez, P. Vanpeperstraete, A. Rueckert, K. Odutola, J. Bennett, L.Tolke, and M. van der Wulp, *ArgoUML User Manual: A Tutorial and Reference Description. Version 0.34 of ArgoUML*. http://argouml-downloads.tigris.org/nonav/argouml-0.34/manual-0.34.pdf. Copyright © 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011.

[24] StarUML. *StarUML 5.0 Developer Guide* http://staruml.sourceforge.net/docs/StarUML_5.0_Developer_Guide.pdf.

[25] Altova Mapforce, http://www.altova.com/umodel/editions.

[26] Acceleo. *Guía del Usuario,Versión 3.1.0*. http://www.obeonetwork.com/group/acceleo/page/acceleo-3-1-0-user-guide.

[27] H. Eichelberger, Y. Eldogan, K. Schmid, *A Comprehensive Analysis of UML Tools, their Capabilities and their Compliance*, Software Systems Engineering. Universität Hildesheim. August 2011.

[28] M. Usman, A. Nadeem, *Automatic Generation of Java Code from UML Diagrams using UJECTOR*, Center for Software Dependability, Mohammad Ali Jinnah University, Islamabad, Pakistan. International. Journal of Software Engineering and Its Applications, vol. 3, no. 2, April 2009.

[29] J. Klein, H. Levinson, J. Marchetti, *Model-Driven Engineering: Automatic Code Generation and Beyond*, Technical Report. CMU/SEI-2015-TN-005, Software Solutions Division. http://www.sei.cmu.edu, March 2015.

[30] J. de Lara, E. Guerra, J. Sánchez, *Model-Driven Engineering with Domain-Specific Meta-Modelling Languages*, Software & Systems Modeling, vol. 14, no. 1, pp. 429-459, February 2015.

[31] J. Bettin, *Best Practices for Component-Based Development and Model-Driven Architecture*, Version 1.0, White Paper, SoftMetaWare Ltd. August 2003.

[32] A. Jakimi and M. El Koutbi. *An Object-Oriented Approach to UML Scenarios Engineering and Code Generation*, International Journal of Computer Theory and Engineering, vol. 1, no. 1, April 2009.

[33] D. Haywood, http://www.theserverside.com/news/1365166/MDA-Nice-idea-shame-about-the. The server Side. Your Enterprise Java Community, 2004.

[34] P. Guha, K. Shah, S. Shankar, P. Shukla, S. Singh, *Incorporating Agile with MDA Case Study: Online Polling System*, International Journal of Software Engineering & Applications (IJSEA), vol. 2, no. 4, DOI: 10.5121/ijsea.2011.2408 83, October 2011.

[35] S. L. Jim, *From UML Diagrams to Behavioural Source Code*, Thesis Universiteit van Amsterdam, Centrum voor Wiskunde en Informatica, 2006.

[36] I. Lipkin and A. K. Huber, *UML Design and Auto-Generated Code: Issues and Practical Solutions*, The Journal of Defense Software Engineering, November 2005.

[37] M. Picón, C. Maldonado, *Generación Automática de Código Basada en Modelos UML*, Cuarta Conferencia Nacional de Computación, Informática y Sistemas (CoNCISa 2016), pp. 134–138, Caracas, Venezuela, 2016.

[38] J. Whitten, L. Bentley, *Systems Analysis and Design Methods*. 7th edition, McGraw-Hill, 2007.

Revista Venezolana de Computación
ISSN: 2244-7040
http://www.svc.net.ve/revecom
Vol. 4, No. 1, pp. 34-46, Junio 2017
Fecha recepción: 28/04/2017, Fecha aceptación: 07/10/2017

Reconocimiento de Patrones Arrítmicos en Registros de Electrocardiografía Dinámica (Holter 24 Horas)

Valentina Colmenárez[1], Esteban Álvarez[2], Robinson Rivas[1], Federico Moleiro[3], Ana Elisa Rodríguez[3]
valentinacc@gmail.com, esteban.alvarez@ciens.ucv.ve, robinson.rivas@ciens.ucv.ve

[1] Escuela de Computación, Universidad Central de Venezuela, Caracas, Venezuela
[2] Escuela de Física, Universidad Central de Venezuela, Caracas, Venezuela
[3] Sección de Cardiología Experimental, Universidad Central de Venezuela, Caracas, Venezuela

Resumen: El objetivo del presente trabajo es el diseño e implementación de un software a partir de herramientas open source, capaz de reconocer patrones de eventos de Fibrilación Auricular Paroxística (FAP) y Taquicardia Ventricular No Sostenida (TVNS), a partir de registros de series temporales - Electrocardiograma (ECG), partiendo de dos parámetros únicos de entrada en cada caso: la irregularidad de los intervalos RR acompañado de la ausencia de la onda P, rasgo característico de la FAP, y el ancho del complejo QRS mayor a 120 ms unido a una frecuencia cardíaca superior a 100 latidos/minuto (lpm), para el patrón de TVNS. Los registros utilizados fueron proporcionados por la Sección de Cardiología Experimental - Instituto de Medicina Tropical - UCV y la $MIT-BIH$ "Normal Sinus Rhythm Databases" de *PhysioNet*. Se realizó un procesamiento inicial a las series temporales a partir del algoritmo de Pan-Tompkins, con el fin de detectar parámetros a estudiar tales como: duración de los intervalos RR, onda P, ancho QRS y frecuencia cardíaca. Para el reconocimiento final de los patrones, a partir de los parámetros citados, se implementó un método para la toma de decisiones, basado en sentencias condicionales, evitando así el uso de un método de decisión más complejo como el de redes neuronales artificiales. Como resultado relevante se obtuvo un detector con: una sensibilidad de 100%, especificidad de 99,99% y exactitud de 99,99% para la detección del patrón de TVNS y para los eventos de FAP una sensibilidad de 75%, especificidad de 99,89% y exactitud de 99,89%.

Palabras Clave: Electrocardiograma (ECG); Complejo QRS; Onda P; Arritmia; Fibrilación Auricular Paroxística (FAP); Taquicardia Ventricular No Sostenida (TVNS); Algoritmo de Pan-Tompkins.

Abstract: We used open source tools for developing a software in order to recognize patterns of Paroxysms Auricular Fibrillation (PAF) and Non-Sustained Ventricular Tachycardia (NSVT) from electrocardiogram time series records. We know the most important features that characterize both cases. The irregularity of the RR intervals and the absense of P wave are features very representative of a PAF case. On the other hand, the width of the QRS complex greater than 120 ms and a heart rate greater than 100 beats per minute (bpm) are features representative of a NSTV pattern. The records were delivered by the Experimental Cardiology Section - Tropical Medicine Institute - UCV and the MIT-BIH "Normal Sinus Rhythm Databases of PhysioNet". We processed the time series with the Pan-Tompkins algorithm in order to detect parameters such as: duration of the RR intervals, QRS width, P wave and heart rate. For the final recognition of the patterns, we used a method to make decisions from above parameters. This method is based on conditional sentences which avoid the use of a more complex decision method such as artificial neural networks. As a relevant result, we got a detector with sensibility of 100% , specificity of 99.99% and accuracy of 99.99% for a NSVT pattern detection and sensibility of 75%, specificity of 99.89% and accuracy of 99.89% for a PAF pattern.

Keywords: Electrocardiogram (ECG); QRS Complex; P Wave; Arrythmia; Paroxysmal Auricular Fibrillation (PAF); Non-Sustained Ventricular Tachycardia (NSVT); Pan-Tompkins Algorithm.

I. Introducción

Las arritmias cardíacas son alteraciones del funcionamiento eléctrico normal del corazón. Algunas son potencialmente malignas y constituyen, con otras enfermedades cardiovasculares, una de las principales causas de mortalidad en el mundo. Según la Organización Mundial de la Salud (OMS) [1], para el año 2012 representaron un 31%, y ese mismo año, sólo en nuestro país un 20,58%, siendo las enfermedades cardiovasculares la primera causa de muerte en Venezuela [2].

La detección de cardiopatías como la Fibrilación Auricular

Paroxística (FAP) y Taquicardia Ventricular No Sostenida (TVNS) es de vital importancia, ya que su detección temprana puede ayudar a establecer estrategias de tratamiento que coadyuven a disminuir el nivel de morbilidad o mortalidad de la población.

Es bien conocida la importancia de la inspección de la señal electrocardiográfica (ECG), de manera visual o mediante técnicas automáticas, para el diagnóstico de una gran variedad de enfermedades cardíacas. El ECG es el registro de la actividad eléctrica del corazón durante un período de tiempo, usando electrodos colocados en posiciones específicas sobre la piel en el tórax del paciente. Este registro presenta aspectos morfológicos intrínsecos de la actividad cardíaca, tal como el complejo PQRST (onda: P, Q, R, S y T; complejo QRS; intervalo: P-Q, Q-T; segmento: ST), ver Figura 1, siendo estos aspectos en una primera instancia los rasgos básicos presentes en el ECG para la identificación de arritmias.

Desde el punto de vista médico, la posibilidad de disponer de herramientas que, a partir del ECG, puedan enfatizar la actividad Auricular y Ventricular, son realmente interesantes y además hoy en día necesarias, pues permiten el desarrollo de técnicas no invasivas de ayuda al diagnóstico clínico, como por ejemplo, clasificación entre distintas manifestaciones de Fibrilación Auricular y Taquicardia Ventricular, así como la detección de otras patologías.

En la actualidad, existen una diversidad de estudios que hacen uso de herramientas computacionales cada vez más complejas y que requieren en cierta medida de un costo de tiempo computacional.

En el mercado existen aplicaciones capaces de identificar tipos de arritmias cardíacas de forma rápida, pero a costos elevados, debido a que se debe adquirir la licencia del producto, o el servicio técnico en caso de presentarse alguna eventualidad, y en algunos casos equipo de hardware especializado.

Ahora, si bien es evidente que estas herramientas de cómputo presentan resultados relevantes, el disponer de herramientas de software que se soporten en el uso de los elementos básicos inherentes a los casos bajo estudio, con estrategias sencillas de procesamiento que ayuden a disminuir los tiempos de cómputo, facilitando además el manejo y entendimiento del propio software, también es importante. En este principio se basa nuestra propuesta, en el manejo de los elementos básicos que conlleven a resultados también relevantes, evitando de esta manera los métodos tradicionales de reconocimiento de arritmias, que pueden ser complejos y tediosos, ocasionando un diagnóstico tardío, que en muchos casos puede ser fatal para el paciente. En este trabajo la combinación de elementos como precisión y rápida detección, como consecuencia de la implementación de técnicas sencillas, es prioritario.

Lo anteriormente expuesto nos motivó a proponer el desarrollo e implementación de una herramienta de software de código abierto para el reconocimiento de estas patologías, que reciba como entrada un registro de ECG. El desarrollo de la herramienta, se abordó en dos etapas: la primera consistió en el estudio y elección de un marco de trabajo para la implementación del software, resultando en el uso de Qt, un framework multiplataforma orientado a objetos ampliamente usado para desarrollar programas que utilicen interfaz gráfica de usuario con el uso del lenguaje de programación C++ de forma nativa; por ser ésta, una herramienta de software libre y código abierto. En una segunda etapa, se abordó el estudio de algoritmos especializados, permitiendo el acondicionamiento de los registros de ECG a partir del algoritmo de Jiapu Pan y Willis J. Tompkins [3], metodología simple presente en la literatura que reune los elementos básicos para el preprocesamiento del ECG, facilitando de esta manera la adquisición de los parámetros mínimos necesarios para la identificación de los patrones asociados a las arritmias específicas bajo estudio. Se desarrolla un algoritmo para la detección de la onda P, complejo QRS y posterior estimación del ancho base del complejo QRS, frecuencia cardíaca instantánea, irregularidad entre intervalos RR, contando con el asesoramiento de la Sección de Cardiología Experimental del Instituto de Medicina Tropical, además de las Escuelas de Física y de Computación de la Universidad Central de Venezuela. Como resultado se desarrolló una herramienta de software que permite el reconocimiento preciso y temprano de eventos arrítmicos tipo FAP y TVNS, a bajo costo ya que las herramientas de desarrollo son de código abierto, para prescindir del costo de licencia.

El trabajo se estructura de la siguiente manera: La Sección II expone los antecedentes de la investigación. La Sección III ofrece una breve reseña de los aspectos cardiológicos mínimos necesarios para entender las anomalías cardíacas tratadas. En la Sección IV se presenta la metodología propuesta para la detección de los patrones arrítmicos. En la Sección V evaluamos la eficiencia del detector de patrones arrítmicos. Posteriormente en la Sección VI presentamos los resultados. Finalmente, en la Sección VII se plantean las conclusiones y en la Sección VIII los trabajos futuros.

II. Antecedentes de la Investigación

Una importante línea de investigación en el área de electrofisiología ha emergido, con el fin de aportar propuestas y herramientas útiles en la detección de patrones arrítmicos, en especial en aquellas arritmias que son de frecuente aparición, tales como la FAP y TVNS, y que muchas veces son desencadenantes de estados aún más críticos, que repercuten de manera negativa en la estabilidad hemodinámica del individuo. Muchas de estas herramientas tienden a ser eficientes en su aplicación, sin embargo, suelen estar diseñadas usando metódos o estrategias que van de moderadamente complejas a complejas y tienden a dificultar su aplicación, por su nivel de abstracción, además generando en algunos casos un alto costo computacional. Comúnmente, las metodologías exploran en el registro del ECG cambios en aspectos morfológicos, temporales, frecuenciales, entre otros, por medio de la evaluación de herramientas comunes tales como: promedio, desviación estandár, pendientes, transformadas de Fourier y ondiculas "Wavelet", hasta otras más elaboradas que usan aprendizaje de máquina (Redes Neuronales Artificiales, Máquinas de Soporte Vectorial), entropía, etc.

En esta sección, se revisan algunos trabajos relevantes relacionados al tema de detección de patrones arrítmicos, contexto de esta investigación:

En [4], se logra la caracterización y reconocimiento de patrones de Taquicardia Ventricular (TV) y Fibrilación Ventricular (FV) por medio de la minería semántica (MS), se añade contenido semántico a los datos y se usan técnicas de Minería de Datos para la extracción de la información de interés. El método alcanzó una alta sensibilidad (Se) y especificidad (Sp) del 96,7% y 98,3%, respectivamente, y fue capaz de detectar el ritmo sinusal normal (N) de las señales TV y FV sin detección falsa, con una Se del 100%.

En [5] se presenta un método para la detección automática y en tiempo real de episodios de FA en ECGs. Este método utiliza intervalos RR, e implica varias operaciones básicas de filtros enteros no lineales / lineales, dinámicas simbólicas y el cálculo de la entropía de Shannon. Usando algoritmos recursivos, se aplica procesamiento analítico en línea. Se logra un rendimiento óptimo con una Se 96,72%, Sp 95,07%, valor predictivo positivo (VPP) 96,61% y precisión global 96,05%, respectivamente.

En [6], la técnica utilizada para identificar los episodios de FAP es a partir del uso de mapas RR vs. dRR, que consiste en un diagrama de dispersión de la frecuencia cardíaca (RR) versus el cambio de la frecuencia cardíaca o de los intervalos RR (dRR). El rendimiento del método se evaluó con registros ECG provenientes de la *MIT-BIH atrial fibrillation database* de *PhysioNet*. Este algoritmo reporta una Se de 90,3% y Sp de 91,2%.

Nidhi y Colaboradores [7], discriminan automáticamente entre el ritmo sinusal normal (N), VT y VF. Para ello, se extrae información discriminatoria de las trayectorias trazadas por las señales N, TV y FV en el espacio de estado. El método de retardo de tiempo se utiliza para representar la señal en el espacio de estado que se convierte a continuación en imagen. En esta imagen, se aplican varias máscaras que clasifican la señal contando el número de píxeles marcados. Se hace uso de la base de datos *MIT-BIH* y se desarrolla en Python 2.7. Los experimentos llevados a cabo dan una precisión del 97%. También el algoritmo desarrollado es computacionalmente menos complejo y por lo tanto puede ser implementado en aplicaciones en tiempo real.

En [8], se presenta un método combinado para lograr una alta precisión en la detección de FA. En primer lugar, se detectaron transiciones sospechosas entre la FA y el ritmo sinusal utilizando la curva de la distribución de diferencias de intervalos RR, que luego fueron clasificados por una combinación de análisis de onda P e intervalo RR. Se utilizó la base de datos *MIT-BIH AF* para la validación de algoritmos y se logró una alta Se y una alta Sp (98,2% y 97,5%, respectivamente). En la práctica clínica se evaluó el rendimiento resultando en una precisión satisfactoria (Se = 96,3%, Sp = 96,8%).

En [9], se presenta el prototipo de monitor ECG para detectar con precisión el ritmo cardíaco irregular, se implementa en un Microcontrolador (MCU) de 8 bits con un detector QRS eficiente usando un algoritmo detector de cambios de pendiente y un algoritmo de detección de arritmia usando la Variabilidad de la Frecuencia Cardíaca (VFC). Este trabajo muestra los resultados de la evaluación del algoritmo en tiempo real utilizando la base de datos Arritmia *MIT-BIH*. La evaluación reportó un 99,72% de Se y 99,19% de predicción positiva. Otro trabajo usando técnicas similares es [10].

En [11], se presenta un estudio comparativo para la detección de FA. Aquí se estudian varios métodos para la detección de FA que se basan principalmente en dos características de los ECG con presencia de FA: la irregularidad de los intervalos RR (IRR) y la actividad auricular eléctrica fibriladora (AAEF). La AAEF se caracteriza por la ausencia de la onda P y la presencia de pequeñas ondulaciones. Nueve algoritmos de detección de FA fueron seleccionados de la literatura y evaluados con el mismo protocolo con el fin de estudiar su rendimiento en diferentes condiciones. Los resultados mostraron que la mayor Se = 97,64% y Sp = 96,08% se logró con métodos basados en el análisis de la irregularidad del intervalo RR, mientras que la combinación de RR y análisis de la actividad auricular dio el valor predictivo positivo más alto (VPP = 92,75%). Los algoritmos basados en la irregularidad de RR fueron también los más robustos contra el ruido (R) (Se = 85,79% y Sp = 81,90% para R = 0dB y Se = 82,52% y Sp = 40,47% para R = -5dB). En este trabajo al igual que en el nuestro se abordan solo los parámetros básicos que caracterizan la FA. Otro estudio similar lo encontramos en [12].

Por otro lado, hay trabajos en los cuales se reconocen y clasifican anomalías cardíacas por medio del uso de métodos de aprendizaje de máquina: Red Neuronal Artificial (RNA), Máquina de Soporte Vectorial (MSV). En [13], emplean en primera instancia el algoritmo de Pan-Tompkins para identificar la onda R. Posteriormente la detección del complejo QRS, la onda P y la onda T son realizadas usando la transformada Wavelet. La clasificación o reconocimiento de las anomalías cardíacas se llevó a cabo con una red neuronal artificial, con retropropagación con gradiente conjugado, escalado de tres capas con 56 neuronas en su capa de entrada (vector de características ECG que se usa de entrada a la red neuronal), 40 neuronas en su capa oculta y 10 neuronas en su capa de salida. Este algoritmo reporta, como resultado de las detecciones los siguientes valores: para latidos normales (N), presentó una Se de 97.0%, precisión de 97.0% positividad de 99,3%. Para anomalías cardíacas de tipo aurícular se obtuvo una Se de 97.7%, precisión de 97.2% y positividad de 92,8% y para las de tipo ventricular se presentó una Se de 98.9%, precisión de 98.3% y positividad de 97,9%. En ese trabajo se usó la Base de Datos de Arritmias *MIT-BIH*.

Así también, Zuluaga et al. [14] presenta la implementación de dos métodos de aprendizaje de máquina: Una red neuronal (RNA) y una máquina de Soporte vectorial (MSV) en un microcontrolador de 32 bits para la detección en tiempo real de la TV y la FV, usando la Transformada Rápida Wavelet para la extracción de características de la señal ECG. La red neuronal tiene una capa de entrada que contiene 4 neuronas,y la capa de salida tiene una sola neurona encargada de clasificar

entre una condición cardíaca normal o una arritmia (TV y FV). Las señales ECG empleadas en el proyecto fueron obtenidas de las bases de datos *MIT-BIH Malignant Ventricular Arrhythmia Database* y *MIT-BIH Normal Sinus Rhythm Database*. Los resultados obtenidos muestran una precisión para la RNA de 99.46% y para la MSV de 99.46%. Otro trabajo similar se encuentra en [15].

Finalmente, en [16] la técnica empleada para detectar la TV consiste en añadir capas de convolución entre las capas de una red neuronal multicapa estándar con retropropagación del error. Esta nueva red se denomina red neuronal convolucional (RNC) y recibe como parámetros de entrada un vector de la señal ECG sin necesidad de pre-procesar previamente. El método propuesto proporciona una medida de Se de 95,6% y Sp de 96,6%, los registros utilizados para evaluar el desempeño, son archivos de *PhysioBank*.

Es importante mencionar que la mayoría de los trabajos existentes en cuanto a la detección de patrones arrítmicos y a los cuales se hace referencia a nivel nacional, han sido derivados del estudio de poblaciones de otras geografías. Este hecho resalta la importancia del presente trabajo donde se emprende el estudio de la detección de patrones arrítmicos empleando datos locales, que muy bien sirven para contrastar comportamientos con datos bajo estudio pertenecientes a otras geografías.

Tomando como base los trabajos anteriormente expuestos que tratan de técnicas o modelos computacionales de reconocimiento de patrones en registros temporales, se propone como objetivo de esta investigación:

A. *Objetivo General*

Diseñar e implementar un software capaz de reconocer patrones de eventos de FAP y TVNS, a partir de registros de series temporales (ECG Holter 24 horas), por medio de herramientas computacionales simples.

B. *Objetivos Específicos*

1) Investigar en la bibliografía los siguientes aspectos: Anatomía y funcionalidad del corazón, electrocardiografía dinámica de Holter 24 horas, arritmias cardíacas.
2) Identificar los cambios característicos que suceden en registros temporales, asociados con los eventos de FAP y TVNS.
3) Por medio del uso de técnicas o metodologías computacionales: implementar, evaluar y entonar el sistema de reconocimiento de patrones propuesto.

III. Aspectos de Cardiología

Se presentan a continuación aspectos básicos de funcionamiento del corazón, en particular se aborda lo inherente al sistema de conducción y arritmias cardíacas.

A. *Sistema de Conducción Cardíaca*

El sistema de conducción según [17], es un sistema especializado que tiene como función, la generación y propagación del impulso rítmico y la contracción coordinada del corazón. El sistema de conducción se divide en dos subsistemas, el primero es un sistema de producción de estímulos, uno con capacidad de automatismo (marcapasos) y el otro permite la conducción de dicho estímulo (ver Figura 1, lado izquierdo).

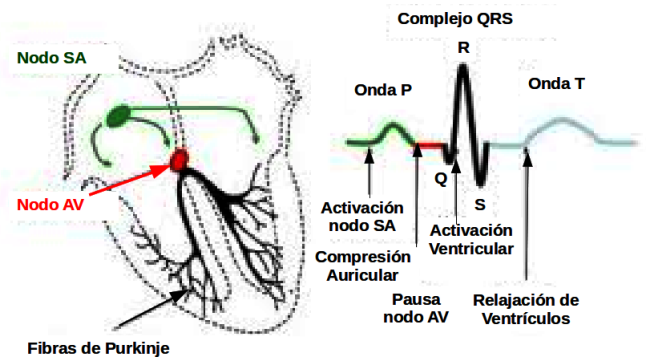

Figura 1: Sistema de Conducción Cardíaca (a la Izquierda) y Electrocardiograma (a la Derecha)

La actividad eléctrica durante cada ciclo cardíaco normal empieza en el nodo sinusal y sigue hasta que todo el corazón se activa. El nodo sinusal, forma parte del subsistema de producción de estímulo, conformado por tejido especializado de conducción con la capacidad de generar potenciales de acción espontáneos, es decir generar el impulso eléctrico. Está situado en la unión entre la vena cava superior y la aurícula derecha, y está compuesto de tejido fibroso llamado fibras nodales o células P, que son el punto de partida para la formación del impulso normal del nodo, a una frecuencia de 60 a 100 latidos/min (lpm), automatismo que supera cualquier otro punto capaz de producir estímulos en el corazón, por esta razón, también es llamado el marcapasos fisiológico [18]. Luego de originarse el impulso eléctrico, este viaja a través de las aurículas, por los fascículos auriculares internodales, permitiendo así la despolarización de la misma, que se representa por la onda P en la señal electrocardiográfica (ver Figura 1, lado derecho), y tiene como punto de llegada el nodo AV, este proceso de despolarización, seguida de repolarización auricular permite el período de llenado y expulsión por parte de las aurículas a los ventrículos. Posteriormente, el nodo AV va a permitir la conducción eléctrica de este impulso a los ventrículos, en forma transversal, a través del haz de His, las ramas derecha e izquierda y las fibras de Purkinje, permitiendo la despolarización de los ventrículos. Esta despolarización se representa a partir del complejo QRS en el electrocardiograma, ulterior a esta despolarización, ocurre un período de repolarización ventricular que es representada en el electrocardiograma por la onda T (ver Figura 1, lado derecho). Este proceso de despolarización y repolarización, permite la sístole y diástole Ventricular, correspondiente al ECG con ritmo sinusal normal (N) (ver Figura 2).

El comportamiento auricular y ventricular ocurre de forma sincronizada. La pérdida de sincronización en ambos procesos

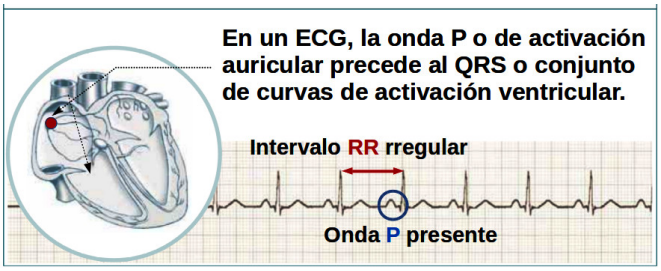

Figura 2: Características de un Ritmo Sinusal Normal

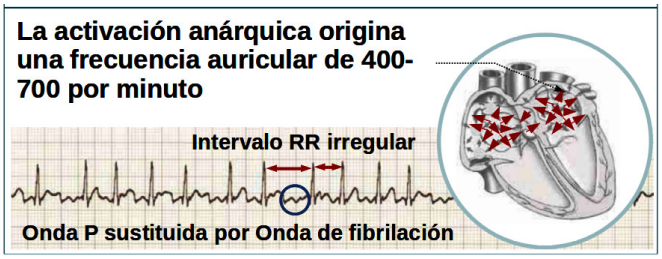

Figura 3: Características de la FAP

trae como consecuencia la aparición de diversas cardiopatías, entre estas, las arritmias que abordaremos en la sección siguiente.

B. Arritmias

La alteración del ritmo cardíaco, consecuencia de la actividad eléctrica anormal en el corazón se conoce como arritmia cardíaca [19]. El estímulo eléctrico originado en el nódulo Sinusal o marcapasos fisiológico, se conduce a las aurículas y los ventrículos de manera regular con una frecuencia que oscila entre 60 y 100 lpm.

En el siguiente apartado, solo se menciona la FA y TV, por ser éstas de mayor interés en este estudio [20].

C. Fibrilación Auricular

La FA es según [21] una arritmia supraventricular caracterizada por una activación incoordinada de las aurículas, (consecuencia de la pérdida de la función hemodinámica de las aurículas) teniendo o no, una respuesta ventricular variable, esto significa que puede presentarse, durante la fibrilación, un comportamiento normal de los ventrículos (QRS normal) o puede generarse una TV (deformación y aumento de la frecuencia de los complejos QRS) al mismo tiempo. Esta arritmia se caracteriza por presentar ondulaciones irregulares de baja amplitud y morfología variable denominadas ondas f, con frecuencia de 350 a 600 lpm, y una onda P ausente; este tipo de arritmia, disminuye el gasto cardíaco en un 25%. En la práctica, se encuentra la FA del tipo Paroxística (FAP), con episodios que se revierten con tratamiento o por sí solos en menos de 72 horas, del tipo persistente con una duración de semanas, y finalmente del tipo permanente, que prevalece en el tiempo.

Las condiciones que debe cumplir una cardiopatía para considerarse FAP (ver Figura 3), son las siguientes:

- Ausencia de la Onda P.
- Como mínimo al menos tres intervalos RR consecutivos diferentes, en 20 latidos contiguos.
- Frecuencia cardíaca superior a 350 latidos por minuto (lpm).

D. Taquicardia Ventricular

La TV se define según [22] como la ocurrencia de tres o más extrasístoles ventriculares (latidos prematuros que se originan fuera del nodo Sinusal) consecutivas con una duración mayor a 120 ms y una frecuencia que varía de 70 a 250 lpm. Se origina en el sistema especializado de conducción, situado en la localización distal a la bifurcación del haz de His, en el músculo ventricular o en combinaciones de ambos tipos de tejido. Los mecanismos suelen ser trastornos en la formación y en la conducción del impulso. Se encuentran TV del tipo No Sostenida (TVNS) que tienen un origen repentino, cuya duración es menor a 30s y Sostenida cuya duración es mayor a 30s; cuando los contornos del QRS son fijos es una Taquicardia Ventricular Monomorfa, si varían al azar Taquicardia Ventricular Polimorfa, si la variación es más o menos repetitiva se denomina Taquicardia Ventricular Helicoidal.

Las condiciones que se deben cumplir para considerar un evento como TVNS (ver Figura 4), son las siguientes:

- Ancho del complejo QRS mayor a 120 ms.
- Poca irregularidad de los intervalos R-R.
- Frecuencia cardíaca mayor a 100 latidos por minuto (lpm).

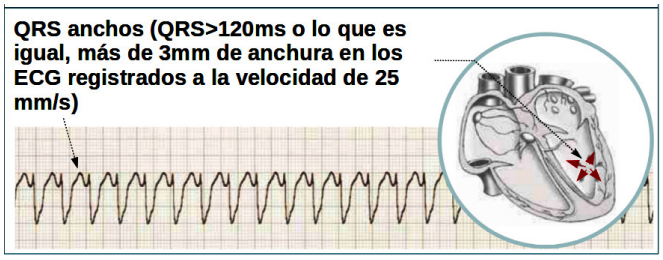

Figura 4: Características de la TV

IV. Metodología

A continuación se describirán los diferentes procedimientos usados en la identificación de los patrones arrítmicos. En una primera instancia se procedió a seleccionar los registros de prueba provenientes de la base de datos de Cardiología Experimental-UCV y la base de datos internacional *PhysioNet*. Luego se procedió a elaborar un algoritmo de extracción de características. Posteriormente se llevó a cabo la detección de los parámetros característicos en cada uno de los complejos PQRST presentes en los registros de prueba. Finalmente en una última instancia se procedió a la fase de identificación en el registro temporal de los patrones arrítmicos (conjunto de complejos PQRST que satisfacen las condiones que definen las anomalías cardiacas estudiadas). El esquema de la metodología planteada se presenta en el diagrama de la Figura 5.

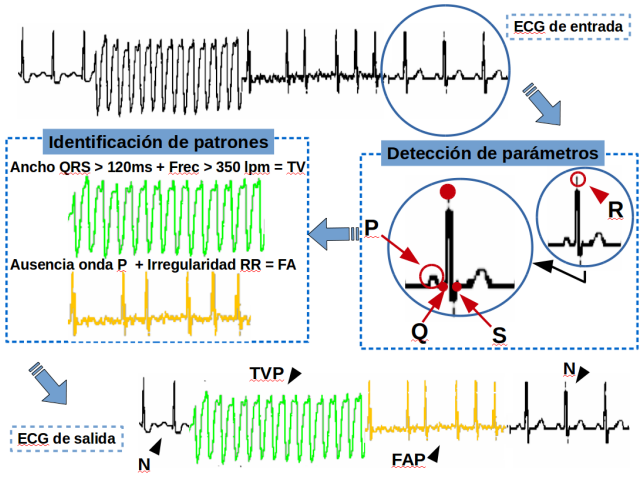

Figura 5: Estructura del Proyecto

A. Selección de Registros de Prueba

Se eligieron al azar para este estudio 36 registros de una hora de duración, los cuales fueron extraídos de registros de Holter de 24 horas provenientes de dos Bases de Datos (BD): Instituto de Medicina Tropical-Sección de Cardiología Experimental-UCV (*IMTCE-UCV*) y *PhysioNet*. Estos registros se dividen en tres grupos: 12 registros presentan al menos un patrón de FAP, 12 registros presentan al menos un patrón de TVNS, ambos grupos provenientes de la BD de *IMTCE-UCV* y 12 registros control (Sanos) sin presencia de anomalías cardíacas, grupo proveniente de la BD de *PhysioNet*. Todos los registros fueron certificados por los médicos especialistas.

Es importante resaltar que se eligieron 12 registros al azar de cada grupo de los casos bajo estudio (FAP, TVNS y SANOS), estimando que estos registros proporcionen un catálogo representativo del ritmo cardíaco en cada caso.

B. Detección de los Parámetros Característicos

El algoritmo para la detección de los parámetros característicos se desarrolló en el lenguaje de programación C++, utilizando programación orientada a objetos a través del *Framework Qt* [23], por ser un entorno de desarrollo de código abierto multiplataforma, y el enfoque de desarrollo de software *Extreme Programming (XP)* [24].

Para la identificación del patrón correspondiente a cada arritmia bajo estudio, fue necesario hallar primero la posición de la onda R asociada a cada complejo PQRST del ciclo cardíaco, tomando como referencia el algoritmo de Jiapu Pan y Willis J. Tompkins [3], el cual se basa en el análisis de la pendiente, la amplitud, y el ancho de dichos complejos, por ser esta onda R, el punto de mayor voltaje de todo el complejo PQRST, y por ende la mejor referencia visual para calcular la frecuencia cardíaca, los intervalos RR, y detectar otras ondas del complejo. Una vez encontrada la posición de la onda R, se procede a localizar la posición de las ondas Q, S y P de cada complejo, usando como referencia, las características de los registros de la señal filtrada, derivada, cuadrada e integrada. Finalmente, a partir de estos parámetros primarios se realizó el cálculo del ancho del complejo QRS, además de la irregularidad temporal entre los intervalos RR, estimación de la frecuencia cardíaca instantánea y la detección de la onda P, haciendo uso de los criterios dados por los médicos especialistas en electrofisiología.

Para detectar los parámetros característicos, se requiere acondicionar la señal en una primera instancia, con el fin de mejorar la relación señal-ruido, facilitando la observación de los mismos. Este acondicionamiento consiste en aplicar al ECG un proceso de **filtrado pasa-banda** para eliminar el ruido de altas frecuencias, como el ruido muscular y la interferencia de la onda T, y obtener sólo la banda en frecuencia donde se esperan encontrar los complejos QRS. El filtro recursivo (pasa-banda) fue propuesto por Pan y Willis J. Tompkins [3] y la ecuación diferencial correspondiente se indica a continuación por medio de la Ecuación 1 (Filtro Pasa-bajos) y Ecuación 2 (Filtro Pasa-altos):

$$y(nT) = 2y(nT-T) - y(nT-2T) + x(nT) - 2x(nT-6T) + x(nT-12T) \quad (1)$$

$$y(nT) = 32x(nT-16T) - [y(nT-T) + x(nT) - x(nT-12T)] \quad (2)$$

los $x(nT)$ representan cada punto o valor de voltaje en el registro ECG.

El siguiente paso en el procesamiento, es un **filtro diferencial** o **derivador** que elimina las componentes de baja frecuencia de las ondas P y T para resaltar las pendientes pronunciadas que caracterizan a la onda R (Ecuación 3):

$$y(nT) = (1/8T)[-x(nT-2T) - 2x(nT-T) + 2x(nT+T) + x(nT-2T)] \quad (3)$$

posteriormente, se aplica una transformación no lineal, que consiste en elevar al **cuadrado** la señal resultante del filtro diferencial, para que todas las muestras sean positivas, acentuando la diferencia entre las distintas pendientes detectadas, observar Ecuación 4:

$$y(nT) = [x(nT)]^2 \quad (4)$$

Por último, se aplica una **ventana móvil integradora**, que promedia las amplitudes elevadas al cuadrado, para eliminar las oscilaciones de poca duración que no corresponden con un complejo QRS, y permite obtener un pulso uniforme en el conjunto de muestras de la señal, asociada a cada uno de los complejos PQRST, a partir de la Ecuación 5:

$$y(nT) = (1/N)[x(nT-(N-1)T) + x(nT-(N-2)T) + \cdots + x(nT)] \quad (5)$$

donde N es el número de muestras en el ancho de la ventana integradora.

Es importante mencionar que posterior a la etapa de filtrado, debe aplicarse sobre la señal una regresión lineal, para eliminar el offset (nivel de corrimiento o desplazamiento)

generado en el proceso de filtrado. Finalmente, luego del acondicionamiento de la señal, se procede a la detección de la onda R. Primero, realizando la búsqueda de un pico de amplitud máxima sobre la señal integrada, esto ocurre cuando se detecta un cambio de pendiente de positivo a negativo y se mantiene por un número prefijado de muestras, luego, a partir del valor de umbrales de amplitud establecidos, propuestos en el algoritmo de Pan y Tompkins [3] se determina si el pico corresponde efectivamente a la onda R, o si debe ser considerado ruido.

Los umbrales (Ecuaciones 6 y 7) son estimadores del nivel de los picos de la señales asociadas a los complejos QRS y del nivel de ruido de la misma, entendiéndose como ruido a cualquier pico de la señal que no sea un complejo QRS. Estos estimadores se actualizan cada vez que se encuentra un nuevo complejo (onda R) a través de un promedio ponderado entre el último valor medio y el valor tomado en la última detección.

$$SPK_I = w_{pk} * PEAK_I + 0,875 * SPK_I \quad (6)$$

$$NPK_I = w_{pk} * PEAK_I + 0,875 * NPK_I. \quad (7)$$

La amplitud del último pico detectado, sea de señal o de ruido, es usado para adaptar el valor umbral a los cambios. Para decidir si se ha encontrado o no un pico a partir de los valores de la señal cuadrada, se obtiene del valor de las pendientes, asegurando con este valor, que el latido corresponde a la onda R y no a una onda T de pendiente pronunciada (ver Ecuación 8). Para realizar un mejor ajuste de los umbrales de amplitud, se toma en cuenta en el algoritmo después de las primeras detecciones, el promedio de los últimos 8 picos R encontrados.

$$TH_1 = NPK_I + w_{th} * (SPK_I - NPK_I) \quad (8)$$

La detección de las ondas Q y S se realiza a partir de la señal derivada. Para el caso de la onda Q, se busca, antes de la posición de la onda R, la ubicación donde se da un cambio de signo, es decir, el valor de amplitud en la señal derivada menor a cero, luego se ubica con mayor precisión la amplitud mínima en torno (ventana de x ms) a la posición donde ocurre el cambio de signo, siendo este último valor, la posición y amplitud de la onda Q. Para detectar la onda S, se lleva a cabo el mismo procedimiento, pero la búsqueda del cambio de signo se realiza a partir de la onda R hasta x ms después de ésta, usando el mismo criterio.

La búsqueda de la onda P, se realiza en el segmento P-Q, a partir de la señal filtrada y la señal derivada, localizando el punto de mayor amplitud y el cambio de signo (criterio utilizado para la búsqueda de las ondas Q y S). Después de hallar el punto de origen y fin de la onda, se calcula el ancho y si es menor o igual a 100 ms, se calcula el máximo de la pendiente a partir de la señal cuadrada, en caso de que esta pendiente sea menor a un porcentaje de la pendiente de la onda R, se ha encontrado una onda P.

C. Identificación de Patrones

La identificación de patrones, inicia con la detección del **ancho del complejo QRS**. Luego de obtener las posiciones de Q y S, en la fase anterior, se calcula la diferencia entre la posición de la onda S menos la posición de la onda Q, para cada complejo, y esta diferencia se multiplica por el tiempo de muestreo, si el resultado es menor a 120 ms éste se considera un complejo QRS normal, en caso de tener un ancho mayor a 120 ms, nos encontramos con un complejo anómalo, ya que la deformación del complejo QRS, indica una falla de la actividad ventricular.

Para llevar a cabo la detección de la **irregularidad de los intervalos RR**, se deben introducir algunas definiciones. Un intervalo RR, es el tiempo que transcurre desde la aparición de un latido u onda R al siguiente, y un intervalo se considera irregular, cuando la diferencia entre dos intervalos RR adyacentes, es mayor a 50 ms. En nuestra implementación, luego de detectadas las ondas R, se calculan los intervalos RR restando la posición entre un latido y el anterior, el resultado se multiplica por el tiempo de muestreo, para conocer su duración en tiempo real. Luego de calculado cada intervalo, se realiza la diferencia entre dos RR consecutivos, si esta diferencia es menor a 50 ms se registra dicho intervalo como regular, en caso contrario, se indica que es irregular.

De esta manera, si el registro contiene más de tres latidos consecutivos con un ancho en cada complejo QRS mayor a 120 ms, y una frecuencia superior o igual a 100 lpm, estamos ante el patrón de la TVNS como se presenta en el diagrama en la Figura 6. Ahora, si el registro presenta más de 3 latidos consecutivos con ausencia de la onda P e irregularidad en sus intervalos RR, estamos ante un patrón de FAP, ver diagrama en la Figura 7. Por último, el criterio para la identificación de un registro de paciente sano, toma en cuenta la presencia de complejos PQRST con intervalos RR que muy bien pueden ser regulares o irregulares, con ancho de cada uno de sus complejos QRS menor a 120 ms y presencia regular de la onda P .

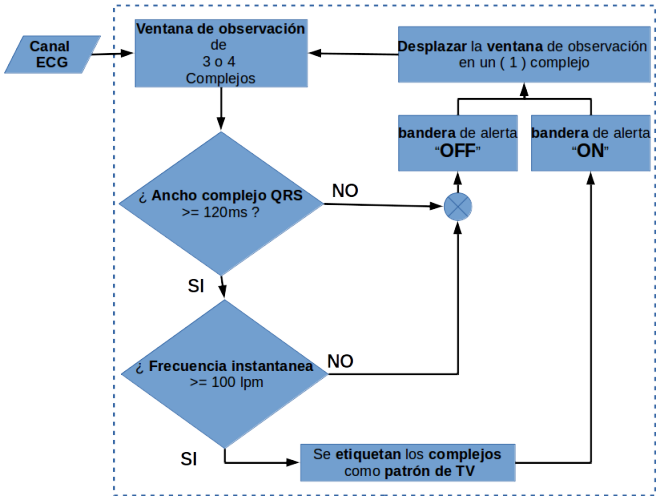

Figura 6: Criterio para Detectar un Patrón de TVNS

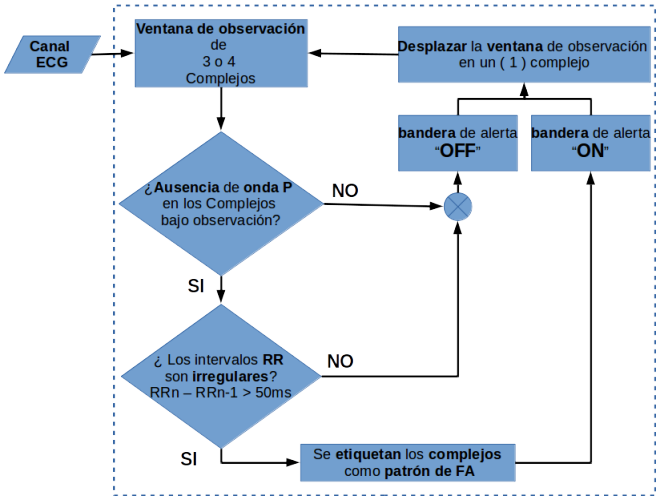

Figura 7: Criterio para Detectar un Patrón de FAP

V. Medida de la Eficiencia del Detector de Patrones Arrítmicos

Una vez desarrollado el software, se procedió a realizar pruebas diagnósticas para establecer la eficiencia del prototipo detector de patrones arrítmicos.

La eficiencia en una prueba diagnóstica es la capacidad para lograr una medición adecuada de la enfermedad, con la finalidad estadística de diferenciar un grupo de pacientes (o muestras a evaluar) con determinada característica o enfermedad de otro grupo sin esa enfermedad. La prueba ideal sería aquella que fuese positiva en todos los enfermos, y negativa en todos los no enfermos.

La sensibilidad y la especificidad son dos valores de probabilidad que cuantifican la fiabilidad diagnóstica de una prueba; para establecer estas cantidades, se toma una muestra suficientemente grande de candidatos a padecer la enfermedad y a todos ellos se les aplica la prueba diagnóstica, evaluando a posteriori los siguientes elementos:

- **VP** son los *verdaderos positivos*: número de pacientes enfermos en los que la prueba dio positiva (diagnóstico correcto).
- **FP** son los *falsos positivos*: número de pacientes sanos en los que la prueba dio positiva (diagnóstico incorrecto).
- **FN** son los *falsos negativos*: número de pacientes enfermos en los que la prueba dio negativa (diagnóstico incorrecto).
- **VN** son los *verdaderos negativos*: número de pacientes sanos en los que la prueba dio negativa (diagnóstico correcto).

donde VP + FP + FN + VN es igual al número de pacientes (muestras) sometidos a la prueba.

A partir de estas cantidades podemos estimar:

- **Sensibilidad (Se)**: La sensibilidad de una prueba es la probabilidad de clasificar correctamente a un individuo enfermo, es decir, la probabilidad de que para un sujeto enfermo se obtenga en la prueba un resultado positivo. La sensibilidad es la capacidad de la prueba para detectar la enfermedad, determinando la proporción de pacientes enfermos que obtuvieron un resultado positivo en la prueba diagnóstica. De ahí, que también se conozca como "fracción de verdaderos positivos", ver Ecuación 9.

$$Se = \frac{VP}{(VP + FN)} = \frac{VP}{Total\ de\ enfermos} \tag{9}$$

- **Especificidad (Sp)**: La especificidad de una prueba es la probabilidad de clasificar correctamente a un individuo sano, es decir, la probabilidad de que para un sujeto sano se obtenga un resultado negativo. La especificidad es la capacidad de una prueba para detectar a los sanos. De ahí que también sea denominada "fracción de verdaderos negativos", ver Ecuación 10.

$$Sp = \frac{VN}{(VN + FP)} = \frac{VN}{Total\ de\ No\ enfermos} \tag{10}$$

- **Exactitud:** La exactitud de una prueba diagnóstica se define como la capacidad que tiene de medir aquello que desea medirse. El cálculo de la exactitud del detector indicará el porcentaje de qué tan exacto es el algoritmo en la detección del patrón de FAP ó TVNS. Esto se muestra en la Ecuación 11.

$$Exactitud = \frac{(VP + VN)}{(VP + VN + FP + FN)} * 100 \tag{11}$$

Lo que realmente nos ocupa en la práctica es estimar la probabilidad de que el paciente (muestra) esté sano o enfermo, según que la prueba haya resultado negativa o positiva:

- **Valor predictivo de una prueba positiva:** es la probabilidad de que un individuo padezca la enfermedad (E) cuando la prueba diagnóstica ha sido positiva (P), tal como se muestra en la Ecuación 12.

$$Pr_{+}(E|P) = \frac{VP}{(VP + FP)} \tag{12}$$

- **Valor predictivo de una prueba negativa:** es la probabilidad de que un individuo esté sano (no-E) cuando la prueba diagnóstica ha sido negativa (no-P), como indica la Ecuación 13.

$$Pr_{-}(no - E|no - P) = \frac{VN}{(FN + VN)} \tag{13}$$

Las pruebas sobre los registros que presentan eventos de FAP y TVNS, además de los registros control, se realizaron tomando como muestras a ser evaluadas conjuntos de 3 ó 4 intervalos RR adyacentes (equivalente a 4 ó 5 complejos PQRST o latidos consecutivos), como se observa en la Figura 8. De esta manera, el número de muestras o evaluaciones a realizar sobre un registro ECG que contiene N latidos o complejos PQRST, vendrá dado por N-4 ó N-5, es decir, una muestra o evaluación por cada desplazamiento de la ventana conteniendo el conjunto de latidos adyacentes. En la caracterización e identificación del comportamiento arrítmico de todas las secuencias de 3 ó 4 latidos cardíacos consecutivos, procedemos a desplazar la ventana de observación en un paso equivalente a un complejo PQRST. La coincidencia de un patrón arrítmico bajo la ventana de observación, se corresponde a un hallazgo. Por ejemplo, a lo que denominamos un **VP** de ese tipo de anomalía.

Se desea conocer si los resultados obtenidos están dentro de los estándares de otras series publicadas.

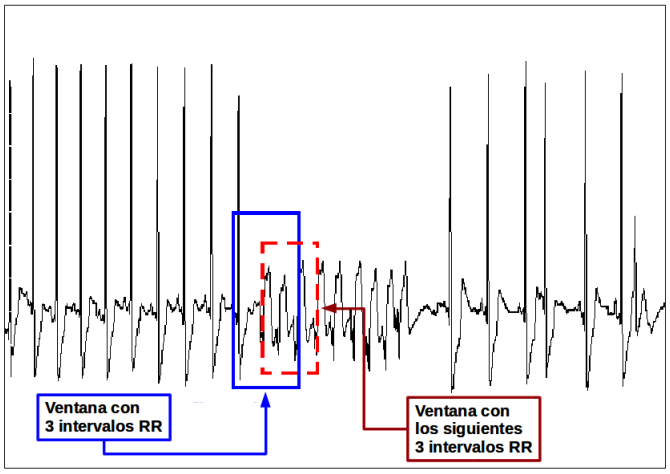

Figura 8: Ventana de Intervalos RR

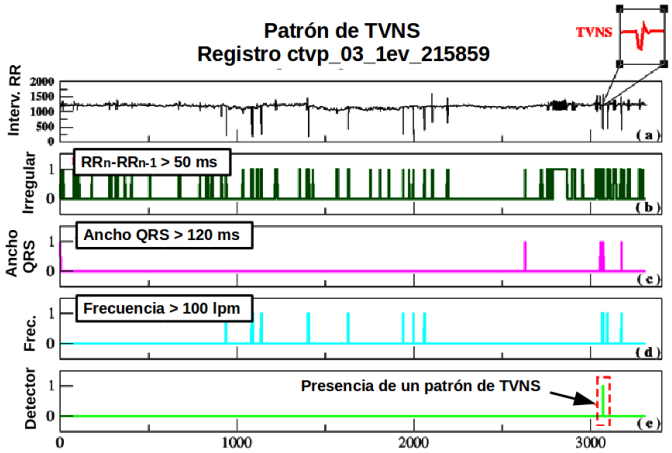

Figura 9: Detección de Patrones de TVNS en BD del *IMTCE-UCV*, con el Esquema de 3 Intervalos RR Consecutivos bajo Observación. (a) Intervalos RR, (b) Irregularidad, (c) Ancho QRS, (d) Frecuencias Instantáneas, (e) Detección de Patrones

VI. Resultados

La fiabilidad de la herramienta se midió a partir del cálculo de algunos indicadores estadísticos como sensibilidad, especificidad y exactitud. Para ésta prueba se han analizado 36 registros: 12 registros presentan al menos un episodio de FAP y 12 presentan al menos un episodio de TVNS, todos provenientes de la BD del *IMTCE-UCV*, además de 12 registros control (sanos), provenientes de la BD Internacional *PhysioNet*, en particular, Instituto Tecnológico de Massachusetts (en inglés, *MIT-BIH Normal Sinus Rhythm Databases*).

A. Registros con Presencia de Patrones de TVNS

Un comportamiento que tiende a ser común en la mayoría de los registros que presentan TVNS, provenientes de la BD del *IMTCE-UCV*, es la presencia de lo que hemos catalogado como la irregularidad, lo cual pone de manifiesto la presencia de complejos adyacentes cuyos intervalos RR difieren en más de 50 ms, ver Figura 9.b donde un 1 indica presencia de irregularidad entre intervalos RR adyacentes y 0 ausencia de irregularidad. También es notable observar algunos complejos cuyo ancho QRS tienden a superar los 120 ms, en la Figura 9.c se indica con un 1 la presencia de esta condición y con 0 la ausencia de la misma. Además, podemos observar frecuencias instantáneas que superan los 100 lpm, ver 9.d donde un 1 indica presencia de frecuencias instantaneas superiores a 100 lpm y un 0 ausencia de esta condición. Estos tres comportamientos señalados nos dan una breve descripción de la dinámica presente en los registros de ECG de aproximadamente una hora de duración que contienen al menos un evento de arritmia tipo TVNS. En específico, cuando nos centramos en el comportamiento mostrado por el ancho del complejo QRS y la frecuencia instantánea, ambos tienden a presentar los valores esperados que caracterizan la presencia de un evento de TVNS justo donde se encuentra el patrón de esta arritmia, es decir, un ancho QRS superior a 120 ms y una frecuencia instantánea que acompaña a estos complejos QRS superior a 100 lpm. De esta manera, cuando cruzamos los observables de ancho y frecuencia, vemos que ambos parámetros en conjunto poseen valores superiores al umbral necesario para identificar la huella o patrón de TVNS, en promedio, con una sensibilidad de 100%, una especificidad de 99,99% y exactitud de 99,99% como se muestra en la Tabla I. Estos resultados nos permiten asociar una alta eficiencia al prototipo detector de TVNS desarrollado haciendo uso de sólo dos parámetros para su detección, un ejemplo de la eficiencia en la detección es mostrado en la Figura 9.e donde un 1 indica la presencia de un patrón de TVNS.

A pesar de la irregularidad que muestran la mayoría de los registros analizados, que de alguna manera dificulta la detección de los intervalos RR y en consecuencia la detección de los anchos y la frecuencia instantánea, la herramienta computacional detecta los patrones de TVNS con una alta eficiencia.

Tabla I: Detección de Patrones de TVNS, con el Esquema de 3 Intervalos RR Consecutivos bajo Observación. Registros Provenientes de la BD del *IMTCE-UCV*

Registro	VP	VN	FN	FP
ctvp_01_1ev_103433	1	5661	0	1
ctvp_02_6ev_230144	1	4328	0	0
ctvp_03_1ev_215859	1	3050	0	0
ctvp_04_1ev_135554	1	4681	0	0
ctvp_05_2ev_054355	1	4753	0	0
ctvp_06_1ev_053000	1	5395	0	0
ctvp_07_1ev_141211	1	4432	0	0
ctvp_09_3ev_141248	1	5047	0	1
ctvp_09_1ev_203803	1	6216	0	2
ctvp_12_1ev_014152	1	3730	0	0
ctvp_19_1ev_180823	1	5779	0	0
ctvp_25_3ev_234713	1	4998	0	2
Total	12	58070	0	6

Resultado de la TVNS	
Exactitud (%)	99,99
Sensibilidad (%)	100,00
Especificidad (%)	99,99

B. Registros con Presencia de Patrones de FAP

Como en el caso anterior procederemos a observar el comportamiento que presentan registros en los que ocurre al

menos un evento de FAP, provenientes de la BD del *IMTCE-UCV*, evaluando el comportamiento de estimadores como: la irregularidad y ausencia de onda P.

La mayoría de los registros a lo largo de la hora tienden a presentar una alta irregularidad (intervalos RR entre complejos adyacentes que difieren en más de 50 ms) como se observa en la Figura 10.b, además de interrupciones y activaciones sucesivas de la onda P en complejos no adyacentes (ver Figura 10.c), acompañados de frecuencias instantáneas entre 50 y 200 lpm (ver Figura 10.d).

Un evento de FAP se identifica cuando coinciden simultáneamente la ausencia de la onda P en 3 complejos (PQRST) consecutivos, con la irregularidad asociada a los intervalos RR en estos, es decir, si encontramos asociado a cada complejo PQRST una diferencia entre los intervalos temporales RR que lo antecede y lo precede mayor a 50 ms, con la ausencia de la onda P (complejo QRST), como mínimo en 3 complejos consecutivos, se alcanza la condición de la activación de un evento de FA, como se aprecia en la Figura 10.e. Cuando empleamos el criterio de detección de

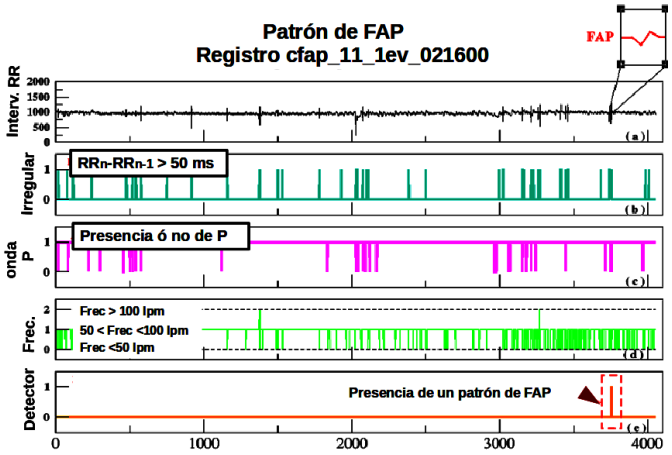

Figura 10: Detección de Patrones de FAP en BD del *IMTCE-UCV*, con el Esquema de 3 Intervalos RR Consecutivos bajo Observación. (a) Intervalos RR, (b) Irregularidad, (c) Onda P, (d) Frecuencias Instantáneas, (e) Detección de Patrones

un patrón de FAP a partir de la presencia de 3 intervalos RR consecutivos irregulares, acompañados de ausencia de onda P, la sensibilidad que se obtiene del detector es de 75%, debido a la detección de falsos patrones de la enfermedad (denominados falsos positivos) en zonas del ECG donde no existe este evento arrítmico (ver Figura 11.e), arrojando una especificidad de 99,89% y una exactitud de 99,89%, como se muestra en la Tabla II. Como se puede observar la sensibilidad obtenida para los casos de la detección del patrón de FAP tiende a ser inferior al resultado encontrado por el sistema detector para los casos de TVNS. De esta manera, nuestro sistema detector de FAP presenta una probabilidad más alta de equivocarse al diagnosticar estos patrones arrítmicos comparado con el detector de TVNS. Sin embargo, es notorio que la especificidad mantiene un nivel muy cercano al 100%.

Ahora si elevamos el nivel de exigencia para la detección de

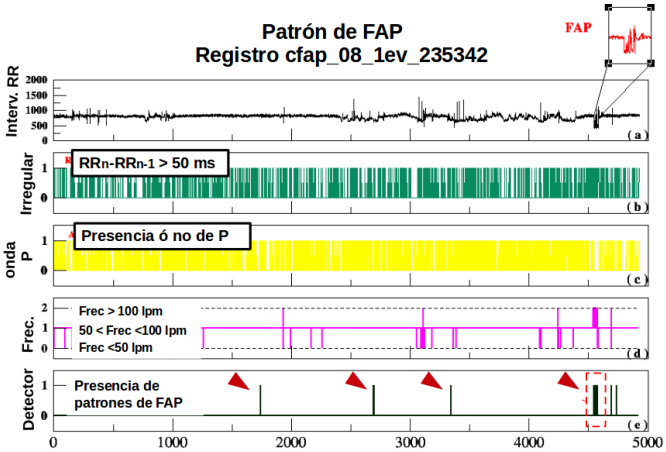

Figura 11: Detección de Patrones de FAP en BD del *IMTCE-UCV*, con el Esquema de 3 Intervalos RR Consecutivos bajo Observación. (a) Intervalos RR, (b) Irregularidad, (c) Onda P, (d) Frecuencias Instantáneas, (e) Detección de Patrones

Tabla II: Detección de Patrones de FAP, con el Esquema de 3 Intervalos RR Consecutivos bajo Observación. Registros Provenientes de la BD del *IMTCE-UCV*

Registro	VP	VN	FN	FP
cfap_03_1ev_195404	1	3462	0	1
cfap_04_1ev_141026	1	4360	0	1
cfap_04_2ev_161829	1	4700	0	0
cfap_08_1ev_235342	1	4552	0	3
cfap_11_1ev_021600	1	3735	0	0
cfap_15_1ev_204006	1	5573	0	3
cfap_22_1ev_113304	1	5604	0	7
cfap_05_1ev_034755	0	5221	1	3
cfap_12_1ev_204610	0	4586	1	16
cfap_13_1ev_232822	0	4168	1	16
cfap_06_1ev_111903	1	4610	0	6
cfap_24_1ev_045002	1	4353	0	2
Total	9	54924	3	58

Resultado de la FAP	
Exactitud (%)	99,89
Sensibilidad (%)	75,00
Especificidad (%)	99,89

FAP usando como criterio involucrar 4 intervalos RR consecutivos irregulares con las respectivas ausencias de la onda P, en lugar de 3, obtenemos como resultado una sensibilidad de 58,33%, una especificidad de 99,99%, y una exactitud de 99,98%, se evidencia una mejora en la especificidad y un deterioro en la sensibilidad (ver Tabla III). Un ejemplo de la mejora de la especificidad al robustecer el criterio se presenta en la Figura 12.e, en la cual se observa que tienden a desaparecer falsos positivos como los observados en la Figura 11.e, incrementando de esta manera la especificidad a expensas de un deterioro en la sensibilidad, ya que se tiende a excluir la presencia de patrones de FAP que involucran 3 intervalos RR irregulares consecutivos.

C. Registros Control

El estudio de los registros control, se lleva a cabo analizando sobre 3 intervalos RR consecutivos los mismos parámetros que han sido observados sobre los registros que presentan eventos de FAP y TVNS. Los indicadores son: irregularidad, frecuencia instantánea, ancho del QRS y presencia de onda P.

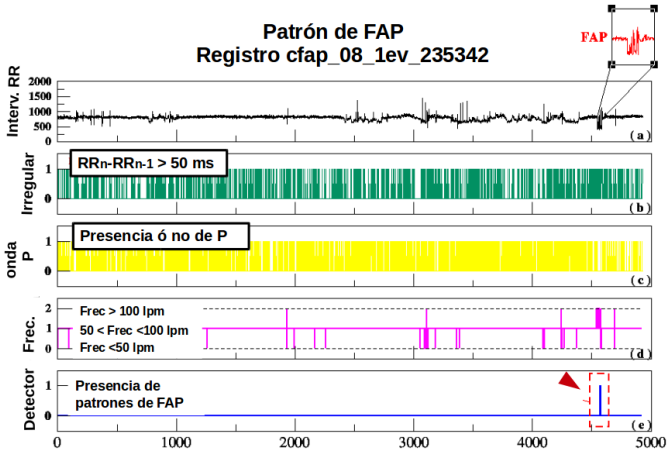

Figura 12: Detección de Patrones de FAP en BD de IMTCE-UCV, con el Esquema de 4 Intervalos RR Consecutivos bajo Observación. (a) Intervalos RR, (b) Irregularidad, (c) Onda P, (d) Frecuencias Instantáneas, (e) Detección de Patrones

Tabla III: Detección de Patrones de FAP, con el Esquema de 4 Intervalos RR Consecutivos bajo Observación. Registros Provenientes de la BD del *IMTCE-UCV*

Registro	VP	VN	FN	FP
cfap_03_1ev_195404	1	3462	0	1
cfap_04_1ev_141026	1	4360	0	0
cfap_04_2ev_161829	1	4700	0	0
cfap_08_1ev_235342	1	4552	0	0
cfap_11_1ev_021600	0	3735	1	0
cfap_15_1ev_204006	0	5573	1	0
cfap_22_1ev_113304	1	5604	0	0
cfap_05_1ev_034755	0	5221	1	1
cfap_12_1ev_204610	0	4586	1	1
cfap_13_1ev_232822	0	4168	1	3
cfap_06_1ev_111903	1	4610	0	1
cfap_24_1ev_045002	1	4353	0	1
Total	7	54924	5	8

Resultado de la FAP	
Exactitud (%)	99,98
Sensibilidad (%)	58,33
Especificidad (%)	99,99

En la totalidad de los casos analizados por medio del prototipo se presenta alta irregularidad (intervalos RR entre complejos adyacentes que difieren en más de 50 ms) como se observa en la Figura 13.b , algunos de estos registros presentan además complejos aislados cuyo ancho QRS tienden a superar los 120 ms (ver 13.c), así como también presencia de interrupciones y activaciones sucesivas de la onda P en complejos no adyacentes aislados (ver Figura 13.d) y frecuencias instantáneas fuera del rango - que corresponden a un ritmo sinusal normal - de 60 a 100 lpm, como se observa en la Figura 13.e. A pesar de las alteraciones observadas (ver Figura 13), consecuencia en la mayoría de los casos de arritmia sinusal respiratoria, común en pacientes que no padecen afecciones cardíacas, el prototipo detector responde de forma efectiva, sin activación de reconocimiento de patrones tipo FAP y TVNS como falsos positivos (ver Figura 13.f), indicando que la dinámica presente en los registros de ECG de aproximadamente una hora de duración pertenecen a pacientes no enfermos, ya que se obtiene una especificidad de 100%.

Este resultado se encuentra además acompañado de una sensibilidad de 100%, ya que hemos detectado un patrón de FAP como verdadero positivo (ver Figura 14.f) por medio del sistema detector en un registro que proviene de la base de datos de pacientes sanos (control). Este hallazgo nos obligó a su certificación por medio de los médicos especialistas quienes confirmaron la veracidad del patrón arrítmico, este patrón no se encuentra identificado dentro de la base de datos de *PhysioNet*. Tal hecho eleva el nivel de confianza sobre el sistema detector al ubicar un evento o patrón, del cual inicialmente no se tenía información.

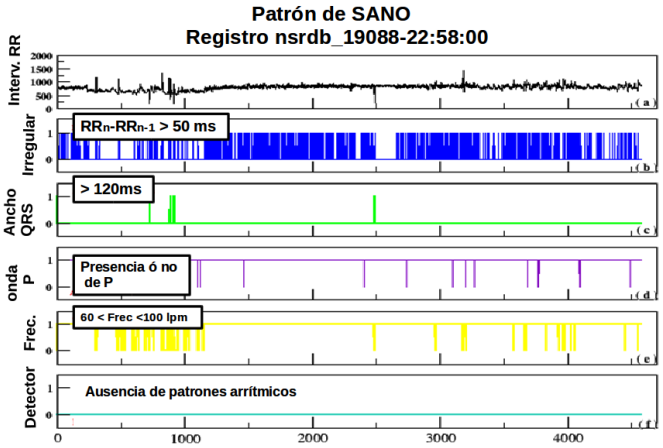

Figura 13: Detección de Patrones de TVNS y FAP en BD de *PhysioNet*, con el Esquema de 3 Intervalos RR Consecutivos bajo Observación. (a) Intervalos RR, (b) Irregularidad, (c) Ancho QRS, (d) Onda P, (e) Frecuencias Instantáneas, (f) Detección de Patrones

En resumen, el desempeño del algoritmo como método diagnóstico en los registros control arroja un resultado altamente eficiente, con una sensibilidad de 100%, especificidad de 100% y exactitud de 100%, como se observa en la Tabla IV.

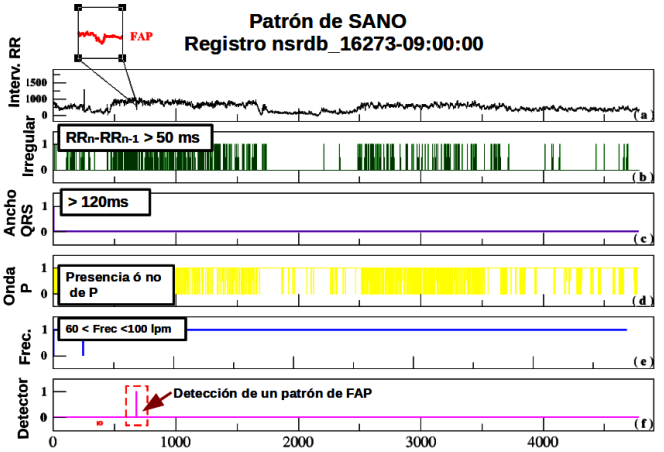

Figura 14: Detección de Patrones de TVNS y FAP en BD de *PhysioNet*, con el Esquema de 3 Intervalos RR Consecutivos bajo Observación. (a) Intervalos RR, (b) Irregularidad, (c) Ancho QRS, (d) Onda P, (e) Frecuencias Instantáneas, (f) Detección de Patrones

D. Resumen

Una vez analizados los diferentes casos bajo estudio, se observó un resultado eficiente de la herramienta prototipo propuesta como detector de los eventos tipo TVNS y FAP

Tabla IV: Detección de Patrones de TVNS y FAP en Registros Control, con el Esquema de 3 Intervalos RR bajo Observación. Registros Provenientes de la BD de *PhysioNet*

Registro	VP	VN	FN	FP
nsrdb-16265-08:04:00	0	460796	0	0
nsrdb-16786-16:48:00	0	460796	0	0
nsrdb-18184-21:34:00	0	460796	0	0
nsrdb-19140-01:39:00	0	460796	0	0
nsrdb-16420-02:55:00	0	460796	0	0
nsrdb-16272-12:45:00	0	460796	0	0
nsrdb-19090-23:50:00	0	460796	0	0
nsrdb-19830-02:56:00	0	460796	0	0
nsrdb-18177-20:30:00	0	460781	0	0
nsrdb-19088-22:58:00	0	460730	0	0
nsrdb-16539-14:40:00	0	460745	0	0
nsrdb-16273-09:00:00	1	460796	0	0
Total	1	5529420	0	0

Resultado R. Control	
Exactitud(%)	100,00
Sensibilidad(%)	100,00
Especificidad(%)	100,00

estudiados, tomando en cuenta un mínimo de parámetros que lo caracterizan para su detección.

Para el caso de la detección de TVNS, la prueba proporciona una sensibilidad del 100% y una especificidad del 99.9%. Además, el valor predictivo positivo ha resultado del 66.7% y el valor predictivo negativo del 100%. Se consideran los resultados aceptables en la detección de este tipo de anomalía cardíaca.

Para el caso de la detección de FAP, la prueba proporciona una sensibilidad del 75% y una especificidad del 99.9%. Con un valor predictivo positivo del 13.4% y el valor predictivo negativo del 99.9%. El valor predictivo positivo es bajo frente a un alto valor predictivo negativo para la detección de FAP.

Es importante resaltar que los resultados obtenidos están dentro de los estándares de otras series publicadas, como se puede apreciar en los trabajos citados previamente, aún considerando que la estrategia empleada se basó en la extracción de información usando metodologías simples.

Sin embargo, no se puede dejar de lado el esfuerzo que se debe hacer en conocer a fondo la dinámica que rige a estos procesos físicos, para poder enfrentar el reto del compromiso de lograr herramientas simples de búsqueda de información, en sistemas que presentan dinámicas altamente cambiantes y complejas en su comportamiento, que dificultan de manera indirecta la detección de los parámetros necesarios para la identificación de los patrones arrítmicos bajo estudio, entre estos elementos podemos citar: cambios de amplitudes, cambios de frecuencia, desplazamiento de la línea base, cambio en la morfología típica del ECG, ruido por interferencias externas, entre otros.

El poder contar con una herramienta confiable en la detección de patrones de arrítmia, le brinda al médico especialista información adicional en la toma de decisiones diagnósticas y/o terapéuticas.

VII. Conclusión

En el presente trabajo se desarrolló un prototipo que permite la detección de patrones arrítmicos tipo FAP y TVNS, hecho que reviste gran relevancia motivado a que este tipo de patrones arrítmicos suelen ser elementos coadyuvantes o predecesores de eventos letales, tales como, infarto o accidente cerebro vascular (ACV) en el hombre.

En la fase de evaluación del sistema de reconocimiento de patrones implementado, se realizaron pruebas sobre un grupo de 36 registros pertenecientes a la base de datos del *PhysioNet* y del Departamento de Cardiología de la UCV. Los resultados obtenidos en la fase de evaluación de la herramienta son una sensibilidad de 100%, una especificidad del 99,99% y exactitud de 99,99% en la detección de eventos de TVNS y para el patrón de FAP una sensibilidad del 75%, especificidad de 99,89% y exactitud de 99,89%, usando ventanas de detección que involucran 3 intervalos temporales RR (4 complejos consecutivos).

De esta manera, entre los objetivos logrados se puede mencionar, que haciendo uso del framework de Qt bajo el lenguaje de programación C++, se pudo llegar al diseño e implementación de un software capaz de reconocer patrones de eventos de FAP y TVNS presentes en registros de series temporales de ECG, a través de la implementación de filtros lineales y no lineales que permiten reducir interferencias y resaltar los cambios característicos de la señal original para luego, a partir de otros métodos computacionales, extraer las características o parámetros que definen cada arritmia, como la onda P y complejo QRS, entre otros, para su posterior análisis.

Es importante mencionar que es posible obtener los parámetros característicos necesarios para el reconocimiento de patrones de las arritmias, a partir de las etapas de procesamiento de la señal, tomando como referencia el algoritmo de Jiapu Pan y Willis J. Tompkins. De esta manera el costo computacional se reduce en comparación con otros métodos como la transformadas de wavelet, redes neuronales artificiales, e inclusive algoritmos más complejos de decisión, permitiendo una detección rápida debido al bajo número de parámetros involucrados y al número reducido de cálculos.

Finalmente, cabe resaltar que es posible obtener resultados eficientes de la herramienta prototipo propuesta comparables a los encontrados en la literatura consultada, como detector de los eventos tipo TVNS y FAP, tomando en cuenta un mínimo de parámetros que lo caracterizan.

VIII. Trabajos Futuros

Como trabajo futuro se propone ampliar las detecciones a otros tipos de patrones arrítmicos y la inclusión de una interfaz gráfica para visualizar los resultados obtenidos durante el desarrollo.

Se sugiere agregar al detector la libre elección del número de complejos consecutivos a ser estudiados, para la detección de los patrones de arritmia. Esta variante permitirá al investigador una evaluación más precisa de los indicadores estadísticos de sensibilidad, especificidad, eficiencia.

Se propone incluir una aplicación web que muestre las estadísticas de los registros ECG de cada paciente e integre los resultados del prototipo detector, además de una aplicación

móvil que permita la consulta de estos datos por el especialista.

Se propone la aplicación de técnicas de minería de datos para generar estadísticas más detalladas de la población bajo estudio. Es necesario un número mayor de registros de ECG para tal fin.

AGRADECIMIENTO

Deseamos agradecer al personal adscrito a la Sección de Cardiología Experimental - Instituto de Medicina Tropical - UCV, por su aporte para el logro de los objetivos planteados en el presente trabajo, y en especial un agradecimiento infinito a la memoria del Dr. Federico Moleiro.

REFERENCIAS

[1] *World Health Organization*, http://www.who.int/en

[2] N. Perez and P. Curcio, *Republica Bolivariana de Venezuela Anuario de Mortalidad 2012*, Ministerio del Poder Popular para la Salud, pp. 6, Septiembre 2014.

[3] J. Pan and W. Tompkins, *A Real-Time QRS Detection Algorithm*, IEEE Transactions on Biomedical Engineering, vol. BME-32, no. 3, March 1985.

[4] M. Othman, N. Safri, I. Ghani, and F. Harun, *Characterization of Ventricular Tachycardia and Fibrillation Using Semantic Mining* Computer and Information Science; vol. 5, no. 5; July 2012.

[5] X. Zhou, H. Ding, B. Ung, E. Pickwell-MacPherson, and Y. Zhang, *Automatic Online Detection of Atrial Fibrillation based on Symbolic Dynamics and Shannon Entropy*, BioMedical Engineering OnLine 2014, 13:18, February 2014.

[6] C. Rotariu, D. Arotaritei, and V. Manta, *Wireless System for Remote Monitoring of Atrial Fibrillation*, IEEE Education and Research Conference (EDERC), 5th European DSP, September 2012.

[7] N. Sharma, A. Cheeran, and P. Gawale, *Automatic Discrimination between NSR, VT and VF*, International Journal of Computer Applications (0975–8887), vol. 93, no. 16, May 2014.

[8] K. Jiang, Ch. Huang, Sh. Ye, and H. Chen, *High Accuracy in Automatic Detection of Atrial Fibrillation for Holter Monitoring*, Univ-Sci B (Biomed & Biotechnol), vol. 13, no. 9, pp. 751–756, September 2012.

[9] A. Lek-uthai, P. Somboon, and A. Teeramongkonrasmee, *Development of a Cost-Effective ECG Monitor for Cardiac Arrhythmia Detection using Heart Rate Variability*, IEEE 2016 Biomedical Engineering International Conference (BMEiCON-2016), Laung Prabang, Laos, December 2016.

[10] S. Mohammad-Taheri, M. Shirazi, and A. Rafiezade, *Slope Analysis Based Methods for Detection of Ventricular Fibrillation and Ventricular Tachycardia*, IEEE 24th Iranian Conference on Electrical Engineering (ICEE), Shiraz, Iran, May 2016.

[11] N. Larburu, T. Lopetegi, and I. Romero, *Comparative Study of Algorithms for Atrial Fibrillation Detection*, Computing in Cardiology 2011, vol. 38, pp. 265–268, September 2011.

[12] I. Marsili, M. Mase, V. Pisetta, E. Ricciardi, A. Andrighetti, F. Ravelli, and G. Nollo, *Optimized Algorithms for Atrial Fibrillation Detection by Wearable Tele-Holter Devices*, 2016 IEEE International Smart Cities Conference (ISC2), Trento, Italy, September 2016.

[13] C. Rose and M. Serna, *Procesamiento del Electrocardiograma para la Detección de Cardiopatías*, Encuentro Nacional de Computación 2014, Ocotlán, Oaxaca, México, Noviembre 2014.

[14] S. Rúa, S. Zuluaga, and A. Redondo, *Machine Learning Algorithms for Real Time Arrhythmias Detection in Portable Cardiac Devices: Microcontroller Implementation and Comparative Analysis*, IEEE XVII Symposium of Image, Signal Processing, and Artificial Vision (STSIVA 2012), Antioquia, Colombia, September 2012.

[15] N. Nuryani, B. Harjito, I. Yahya, and A. Lestari, *Atrial Fibrillation Detection Using Support Vector Machine*, International Conference on Electric Vehicular Technology and Industrial, Mechanical, Electrical and Chemical Engineering (ICEVT & IMECE), IEEE, Surakarta, Indonesia, November 2015.

[16] B. Chandra, C. Sastry, and S. Jana, *Subject-Specific Detection of Ventricular Tachycardia Using Convolutional Neural Networks*, Computing in Cardiology 2016 (CinC), vol. 43, pp. 53–56, September 2016.

[17] G. López, *Semiología Cardiovascular*, 5ta. Edición, pp. 18-19, 1999.

[18] E. Braunwald, *Tratado de Cardiología*, Editorial McGraw-Hill Interamerica, vol. 1, 5ta. Edición, pp. 599, 1999.

[19] V. Fuster y R. Wayne, *HURST: El Corazón*, Editorial McGraw-Hill Interamerica, vol. 1, 10ma. Edición, pp. 767, 2001.

[20] V. Fernández y J. López, *Fisiología del Ejercicio*, Editorial Médica Panamericana, 3ra. Edición, pp. 442-443, 2006.

[21] O. Gutiérrez y V. Araya, *Manual de Arritmias Cardiacas: Guía Diagnóstica Terapéutica*, Editorial de la Universidad de Costa Rica, 1ra. Edición, pp. 11–12, 269p, 2002.

[22] R. Huszar, *Arritmias: Principios, Interpretación y Tratamiento*, Editorial Elsevier, 3ra. Edición, pp. 7-21, 2002.

[23] *Qt-Project.org*, https://www.qt.io/developers

[24] E. Kendall y J. Kendall, *Análisis y Diseño de Sistemas*, Editorial Pearson Educación, 6ta. Edición, 2005.

Revista Venezolana de Computación
ISSN: 2244-7040
http://www.svc.net.ve/revecom
Vol. 4, No. 1, pp. 47-58, Junio 2017
Fecha recepción: 28/04/2017, Fecha aceptación: 10/10/2017

VMAS-Modeller: Una Aplicación Visual para el Modelado de Sistemas Multi-Agentes Guiado por la Metodología MASINA

Sredny Buitrago[1], Manuel Sánchez[1]
sredny.nevermind92@gmail.com, mbsanchez@unet.edu.ve

[1] Departamento de Ingeniería en Informática, Universidad Nacional Experimental del Táchira, San Cristóbal, Venezuela

Resumen: El presente artículo describe un software para el modelado y generación de código de Sistemas Multi-Agentes siguiendo las fases y modelos establecidos por la metodología MASINA El código generado corresponde a la estructura principal de clases que conforman el proyecto que está siendo desarrollado por el usuario, dicho código se ejecutará en la plataforma de agentes JADE; además, incluye una serie de funcionalidades que hacen al software bastante atractivo para desarrolladores y diseñadores de SMA. El principal propósito de la aplicación es generar una estructura básica de clases que integren el proyecto partiendo de los modelos de MASINA: modelo de agentes, modelo de tareas, modelo de comunicaciones, modelo de coordinación y modelo de inteligencia, además permite la exportación del proyecto en formato XML versión 1.0, copiar la descripción de los modelos en el portapapeles para facilitar la creación de documentación, ensayos, artículos académicos, etc. Por otro lado, se utilizó FDD (Desarrollo Guiado por Funcionalidad, en español) como metodología de desarrollo y MDA (Arquitectura Guiada por Modelos) como metodología para la interpretación y generación de código a través de los modelos.

Palabras Clave: Sistemas Multi-Agentes; Herramientas CASE; JADE; MASINA; Generación de Código.

Abstract: This paper describes a software solution for modeling and code generation of Multi-Agents Systems following the steps and models established in MASINA methodology. The generated code is the main structure of the classes that compounds the whole project in JADE platform; also, it involves a series of functionalities that makes the developed tool very attractive for SMA designers. The main purpose of the application is to generate a basic structure of the classes that integrate the project in JADE starting from the models of MASINA, these models are: agents model, task model, communication model, coordination model and intelligence model, also provides some others helpful features as exporting the current project in a XML v-1.0 file and copy the models in the clipboard to generate documentation, papers, academics articles, etc. In other hand, the methodology used in the development of this software was FDD (Feature Driven Development) and MDA (Model Driven Architecture) for the code generation.

Keywords: Multi-Agents Systems; CASE Tools; JADE; MASINA; Code Generation.

I. Introducción

La automatización industrial ha traído consigo la aparición de nuevos retos con respecto a cómo dar solución a cada uno de los problemas de manera eficiente y eficaz, muchos de estos problemas tienen que ver con tiempos de respuestas muy lentos, inadaptabilidad de los sistemas a nuevos estímulos del entorno, autonomía, descentralización, entre otros. A lo largo del tiempo se han propuesto distintos tipos de soluciones que permiten resolver los problemas mencionados de forma computacional; una de ellas son los Sistemas Multi-Agentes (SMA), éstos sistemas han comenzado a acrecentar con fuerza su presencia en el mundo del software debido a las posibilidades que ofrecen, la diversa aplicabilidad y la cantidad de plataformas que los soportan. En Efecto, un SMA puede representar una solución adecuada y óptima de acuerdo a la naturaleza del problema que se afronta, puesto, que éstos sistemas tienen como premisa la división del trabajo en tareas que serán posteriormente ejecutadas por los agentes de forma paralela, de tal forma que se aumenta la eficiencia del sistema proveyendo un tiempo de respuesta aceptable. Por otra parte, el área de diseño y creación de SMA aplicado a la automatización industrial se va expandiendo día a día, debido a que se presenta como solución a problemas de diversa índole y complejidad; Sin embargo, la creación de un SMA es un proceso extenso en el que se deben tomar en cuenta muchos factores [1], por lo que se han diseñado y postulado diversas metodologías para la creación de estos sistemas, entre las que se encuentra la metodología MASINA [2] (MultiAgent Systems for INtegrated Automation), la cual es una propuesta metodológica para la especificación e implementación de sistemas basados en agentes en ambientes de automatización industrial. En consecuencia, MASINA abarca la creación de SMA pasando a

través de 5 fases: conceptualización, análisis, diseño, codificación y pruebas. En cada una de las fases de ésta metodología, se crean diagramas, mediante el llenado de las plantillas que aporta la misma, lo que permite al diseñador del SMA, contar con información importante que permite definir las propiedades, objetivos, tareas, modelos de interacción, comunicación, inteligencia, entre otros, de los agentes del sistema. En algunas ocasiones, resulta confuso para los diseñadores del SMA llenar apropiadamente (con la información adecuada para cada campo) las plantillas que define la metodología, situación que es frecuente entre los usuarios que tienen poco conocimiento de la metodología, lo que puede conllevar al fracaso o al mal uso de la misma, obteniendo como resultado un sistema que no se ajusta a los requerimientos de diseño definidos inicialmente. Como se mencionó anteriormente, el diseño y desarrollo de los SMA es una tarea que requiere dedicación y una gran cantidad de tiempo, por lo que es necesario que sea apoyada a través de herramientas de software especializadas que usen una metodología de diseño y un ciclo de desarrollo adecuados. Esto ha motivado a desarrollar una aplicación CASE [3] (Computer-Aided Systems Engineering) que permita automatizar el proceso de diseño y desarrollo de un SMA. En este sentido, se dice que una herramienta CASE es una aplicación que ayuda en el proceso de desarrollo de software en forma automatizada. En consecuencia, ésta investigación se enfoca en el desarrollo de una aplicación CASE para el diseño de Sistemas Multi Agentes guiados por medio de la metodología MASINA.

La herramienta desarrollada debe contar con la capacidad de generar el código inicial del proyecto para la plataforma de despliegue de agentes JADE [4] (Java Agente Development Framework) a partir de la información captada por medio de los modelos de la metodología MASINA. Por su parte JADE es un middleware que facilita el desarrollo de sistemas multi-agente bajo el estándar FIPA, que cuenta con un entorno de ejecución en el que los agentes co-existen. De esta manera, la creación de la aplicación propuesta en esta investigación va a favorecer la construcción de sistemas Multi-agentes, facilitando la creación de los diagramas de la metodología, al guiar al diseñador paso a paso a través que aporta la metodología, lo que asegura que el SMA sea creado de forma metodológica, y a su vez, facilitar a los usuarios inexpertos el uso correcto de la metodología MASINA. De igual forma, la aplicación permite ingresar la información, de una manera sencilla para el usuario, parametrizando aquellos campos con valores conocidos y validando la información ingresada por el mismo, para asegurar que el código generado sea correcto. Finalmente, la aplicación permitirá generar una gran cantidad de código que beneficiará el tiempo de desarrollo del SMA.

En general, este artículo se estructura de la siguiente manera: En la Sección II se presenta el marco teórico, en la Sección III se muestran los trabajos relacionados, la Sección IV presenta el desarrollo de la aplicación y en la Sección V se define un caso de estudio; en la Sección VI se hace una pequeña experimentación; para finalizar con algunas conclusiones acerca del producto desarrollado.

II. MARCO TEÓRICO

Los sistemas Multi-agentes son parte del paradigma de inteligencia artificial distribuida [5], se trata de soluciones de software que afrontan problemas dividiéndolos como una serie de tareas que serán ejecutadas por los entes o agentes que conforman el sistema. Un agente es esencialmente un componente de software especial que posee autonomía y provee una interfaz interoperable a un sistema principal o que se puede comportar como un agente humano, trabajando para ciertos clientes en función de cumplir su propia agenda [6]. Por tanto, es vital la comunicación entre los diferentes agentes de tal forma que se pueda alcanzar el objetivo de diseño eficazmente, por lo que es necesario contar con un protocolo que coordine la comunicación entre los diferentes entes.

Al ser el desarrollo de estos sistemas un proceso de alta complejidad se han propuesto diversas metodologías para el modelado de SMA, entre ellas tenemos las orientadas a objetos [7-14] y las metodologías basadas en el paradigma de agentes [15-22]. Sin embargo, éstas presentan deficiencias, debido a que dificultan el modelado de partes complejas del sistema mientras que otras permiten inconsistencia con respecto a la información proporcionada. MASCommonKADS, es una de las metodologías de diseño de agentes ampliamente conocidas, la cual surge como una extensión o mejora de la metodología de ingeniería del conocimiento CommonKADS al incluir componentes orientados a objetos (OO) y de la ingeniería del protocolo para la definición de los protocolos del agente [6]. MASCommonKADS se considera como la más completa y adecuada para el modelado de SMAs de automatización industrial, no obstante, ella no toma en consideración los fundamentos inteligentes del agente. Debido a lo anterior, Aguilar et. al proponen MASINA [2] como una metodología de modelado de Sistema Multiagentes supliendo así las carencias que presentan las demás metodologías. En consecuencia, MASINA es una extensión de MASCommonKADS, cuya diferencia primordial es que esta última agrega los modelos para la especificación de la inteligencia del agente, además, MASINA se orienta al modelado de procesos relacionados a la automatización industrial. Aunado a esto, MASINA permite especificar el modelo de inteligencia individual e inteligencia colectiva, así como de establecer las tareas que un agente debe realizar para lograr sus objetivos. Es importante aclarar, que la inteligencia en sistemas multi-agentes se entiende como la capacidad que tiene los agentes para ir tras sus objetivos y ejecutar las tareas de una manera optimizada [5], a la vez que permite detallar lo concerniente a la coordinación y comunicación entre los agentes involucrados en el sistema, es decir, intercambio de mensajes, y métodos para la resolución de conflictos. En consecuencia, la metodología MASINA define 5 modelos para el diseño de todo el sistema, ellos son: modelo de agente, modelo de tareas, modelo de comunicación, modelo de coordinación y modelo de inteligencia (colectiva e individual). Finalmente, cabe mencionar que MASINA agrega nuevos atributos a los modelos existentes en MASCommonKADS, de tal manera que se puede ser más específico en la descripción de los agentes. Entre esos tributos se tiene: componentes del SMA y marco de referencia al modelo de agentes. En tal sentido, en el modelo de tareas se especifican las sub-tareas y cuáles de

ellas hacen uso de técnicas de inteligencia, mientras que, en el modelo de coordinación se definen las conversaciones que se efectúan entre los agentes registrados.

Por tanto, al ser MASINA una metodología que cubre las deficiencias de las metodologías mencionadas anteriormente, y debido a las ventajas que ésta presenta respecto a las demás, se consideró para ser implementada en esta investigación, con lo que se pretende crear un software denominado Visual MAS Modeller (V-MAS) que tiene la capacidad de generar código para el despliegue de Sistemas Multi-Agentes en JADE, a partir de los modelos de la metodología MASINA.

Por otra parte, JADE se presenta como una plataforma, que provee los mecanismos necesarios para que los entes de un SMA puedan coexistir (descubrimiento, comunicación, coordinación, etc.), así como las herramientas para la detección y búsqueda de servicios que ofrecen los agentes [23]. Esta plataforma se caracteriza por su alto uso como plataforma de despliegue de agentes y su orientación al desarrollo de sistemas multi-agentes bajo el estándar FIPA [24]; Esta son las razones por las cuales se considera JADE como plataforma objetivo de Visual MAS Modeller lo que permitirá cubrir una amplia cantidad de usuarios y sistemas. Sin embargo, el sistema ha sido desarrollado con la capacidad de que en el futuro se puedan agregar generadores de código para otras plataformas de despliegue de agentes.

En el mismo orden de ideas, las herramientas CASE son programas de software cuyo objetivo es aumentar la productividad en el desarrollo de aplicaciones computacionales [25], es decir, éstas se encargan de automatizar ciertas áreas del ciclo de vida del software, ahorrando tiempo de desarrollo y, en consecuencia, los costos asociados al mismo. Por definición la ingeniería de sistemas asistida por computador es la aplicación de tecnología informática a las actividades, las técnicas y las metodologías propias del desarrollo de sistemas. El propósito de esta tecnología es acelerar el proceso de desarrollo de sistemas y mejorar la calidad de los sistemas resultantes [25]. En este mismo sentido, se puede decir que hoy en día existe una amplia gama de este tipo de herramientas, que abarcan desde el modelado de sistemas, gestión de tiempos y costos de proyectos; hasta la generación de código para ciertas plataformas específicas, siendo este último el que mayor se adapta a los objetivos de V-MAS. Los generadores automáticos de código sirven como herramientas de apoyo a los desarrolladores, debido a que se puede procesar una gran cantidad de información en cuestión de segundos y ahorrar un trabajo que muy bien puede ser automatizado, produciendo, además, un código libre de errores. De esta forma V-MAS, es capaz de disminuir el tiempo de creación de un SMA, al crear de forma automatizada la estructura principal del proyecto del SMA en cuestión, de tal forma que el desarrollador del SMA pueda centrarse en actividades más específicas y complejas como la implementación de algoritmos de aprendizaje, algoritmos de razonamiento o algoritmos para la resolución de conflictos, entre otros, siendo así como se establece la relación entre VMAS y las herramientas CASE.

III. Trabajos Relacionados

Es indiscutible que no existe una herramienta que reúna las características principales que presenta V-MAS (generación de código para JADE, uso de MASINA como metodología de modelado, exportación de modelos en forma tabular, usabilidad, etc.), se han desarrollado otras herramientas de software similares, que partiendo de algunos datos ingresados por el usuario son capaces de generar código inicial hacia otras plataformas de destino. Éstas herramientas sirvieron como base o referencia para la presente investigación, de tal forma que para lograr que V-MAS sea una aplicación estable, consistente con la metodología y usable se tomaron en cuenta los aspectos mejorables, fallas y debilidades que estos trabajos presentan, entre ellas se tiene: no son multiplataforma, escasa validación de la información ocasionando inconsistencias y errores en el código generado, usabilidad de usuario limitada, lo que indica que son difíciles de usar y de instalar, no generan código para plataformas de gestión de agentes, lo que le agrega mayor carga de trabajo al usuario que debe ocuparse de todos los mecanismos necesarios para que los agentes puedan co-existir, como registro, búsqueda, comunicación, coordinación, entre otro de los agentes; no toman en cuenta a los usuarios inexpertos, lo que hace que la curva de aprendizaje en la creación de SMAs sea más pronunciada, entre otros. Dentro de los trabajos previos se tiene:

EDISMA [25] (Entorno de Desarrollo Integrado para la creación de Sistemas MultiAgentes) desarrollada por Rivero en la Universidad de Los Andes (ULA), ésta herramienta presenta una interfaz gráfica de usuario (IGU) que permite captar la información de los modelos de MASINA, para posteriormente generar el código en lenguaje C++, es decir, que crea archivos que contienen los métodos e instanciación de los agentes en lenguaje C++, así como el archivo makefile para compilar el SMA en cuestión. Por otra parte, Mészáros, de la Universidad Comenius de Bratislava, Eslovaquia en su tesis de maestría titulada code generation from AML (Archetype Modeling Language) to jadex [27], desarrolló un generador de código que parte de los modelos AML para generar el código de los agentes en JADE, dicha traducción de los modelos a código se realizó haciendo uso de las fases que propone la metodología MDE (Model Driven Engineering). Seguidamente, en el año 2009 en La Universidad de Los Andes (ULA) Aguilar y Bravo desarrollaron un proyecto que denominaron SISGECOMA [23] (Sistema Generador de Código para MASINA) cuyo objetivo principal fue diseñar e implementar un sistema traductor de los modelos de MASINA a C++, para lo cual se desarrollaron 3 agentes, uno que se encarga de desplegar la IGU y recolectar la información, otro se encarga de la validación y verificación de la información obtenida y finalmente un agente que realiza la generación de código [28].

V-Mas por su parte guía al usuario paso a paso en el uso de la metodología, permitiendo ingresar y validar los datos paso a paso y de manera sencilla, logrando que el usuario pueda crear un SMA sin necesidad de ser un experto en la metodología, pero asegurando que el desarrollo del mismo sea adecuado al ciclo metodológico de la metodología MASINA, lo que a su vez certifica que se cumplan los objetivos de diseño. De igual manera, V-MAS asegura que el código generado esté libre de

errores de sintaxis, lo que deriva en un ahorro de tiempo en la fase de codificación, a la vez que el usuario no debe preocuparse por implementar algoritmos para la co-existencia del SMA, ya que la plataforma JADE será la responsable de ello y de asegurar que el SMA cumpla los estándares de la FIPA.

IV. Visual MAS Modeller (V-MAS)

V-MAS Modeller es una aplicación que busca facilitar el trabajo de los desarrolladores de SMA basándose en los conceptos, modelos y pasos propuestos en MASINA. Así V-MAS apoya el modelado y desarrollo del SMA, además de facilitar la utilización de la metodología MASINA. De igual manera, V-MAS, es una aplicación de alta usabilidad, al proveer los formularios para la introducción de la información de los modelos de manera secuencial y simple. Por otra parte, V-MAS permite guardar el proyecto en formato XML, lo que aumenta la integración con otras aplicaciones que pueden tomar el modelo como entrada para realizar operaciones sobre el mismo, como, por ejemplo, generar el código para otras plataformas de agentes, generar gráficos que muestren la interacción entre los agentes, etc. V-MAS se presenta como un medio de apoyo para aquellos desarrolladores de SMA que conozcan o no la metodología MASINA, ya que V-MAS hace más fácil y eficiente la implementación e ingreso de la información requerida por los modelos de la metodología, lo que significa un ahorro de tiempo, esfuerzo y costos en la programación. Por otra parte, la integración con diversas aplicaciones de diagramado UML hace que los diseñadores no tengan que abandonar sus IDE's de modelado UML tradicionales, sino que, por el contrario, vienen a ser un soporte sobre las tareas y funciones que se desarrollarán en el sistema. En consecuencia, el propósito de V-MAS Modeller, es apoyar al usuario en el desarrollo de los SMA, basándose en el ciclo de vida la metodología MASINA y de los sistemas multiagentes, a la vez de contar con funcionalidades básicas de un modelador como son: guardar, exportar y generar código, compartir el proyecto con otros usuarios, e importar los diagramas UML desarrollados en modeladores como StarUML y ArgoUML, según se describe en la Figura 1, además V-MAS facilita la exportación de la descripción de los modelos a distintos procesadores de texto como Microsoft Word a través del portapapeles del Sistema operativo (copiar y pegar) esto es una ventaja para los desarrolladores a la hora de crear documentos como informes, artículos para publicaciones, etc. V-MAS Modeller surge como una solución a la inexistencia de modeladores guiados por MASINA y la generación de código para JADE.

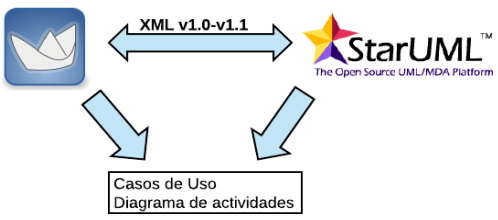

Figura 1: Importación de Archivos XML desde los Modeladores UML

El desarrollo de la aplicación se guió a través de 2 metodologías: MDA (Model Driven Architecture) y FDD (Feature Driven Development). Por un lado, MDA establece la forma en que se traducirá la información introducida por el usuario en los modelos de MASINA a código Java, es decir, instaura la relación entre la información introducida en los modelos y el código que debe ser generado por VMAS. Seguidamente, FDD es una metodología ágil que guía el desarrollo de herramientas de software y gestiona cada una de las funcionalidades a implementarse, ella cubre las etapas clásicas de desarrollo del software, como son: definición de requisitos, diseño y documentación, desarrollo, pruebas y despliegue haciendo uso de iteraciones cortas [29]. Específicamente, las etapas de desarrollo propuestas por FDD [30] son:

- Desarrollo de un modelo general.
- Construcción de una lista de características.
- Plan por característica.
- Diseño por característica.
- Construcción por característica.

Por tanto, con el fin de lograr un software robusto, al momento de listar las características o funcionalidades de V-MAS, se clasificaron en 6 diferentes áreas o módulos, ellos fueron: creación de agentes, creación de modelos, configuración de parámetros de la aplicación, generación de código, generación de documentación (copiar modelos, así como la relación tarea-servicios mediante el portapapeles) y administración del proyecto. Así pues, FDD se trata de una metodología que permitirá el acoplamiento con MDA perfectamente sin descuidar los aspectos relacionados a la calidad del software. Inicialmente el desarrollo de V-MAS es guiado por FDD, y en el momento de desarrollar la característica de generación de código se conmuta a MDA, de esta manera se integran ambas metodologías.

Una de las funcionalidades de V-MAS, es la de proveer las interfaces, formularios y mecanismos adecuados para la correcta y eficiente captura de la información. De esta manera, el ciclo de vida de un proyecto de SMA en V-MAS comienza con el registro de los agentes necesarios, posteriormente para cada uno de ellos se debe ingresar la información correspondiente a los modelos de MASINA tales como: modelo de agentes, modelo de tareas, modelo de comunicación, modelo de coordinación, modelo de inteligencia individual, modelo de inteligencia colectivo; para finalmente proceder con la generación del código. Es importante resaltar que mediante V-MAS no es posible saltarse pasos en la metodología, lo que asegura la consistencia y completitud de la información, por ejemplo, no es posible cargar el modelo de tareas si aún no se ha cargado el modelo de agentes. Esto no se ve como una falla sino como una fortaleza ya que asegura un buen uso de la metodología y la calidad del código generado.

De esta manera, una vez se cuenta con la información provista por el usuario V-MAS procederá a generar la mayor cantidad de elementos de código que sea posible, es decir, clases, métodos, atributos de clases, clases auxiliares y librerías. Para cumplir este objetivo es necesario que V-MAS interprete

correctamente los modelos de MASINA. En consecuencia, La fase de generación de código fue realizada utilizando la metodología MDA, ella propone primeramente realizar un modelo independiente de la plataforma (PIM) seguidamente transformarlo a un modelo específico de la plataforma (PSM) y por último proceder a la generación del código [23]. En la presente aplicación, el modelo PIM está representado por los modelos de MASINA ingresados por el usuario en V-MAS (V-MAS permite la especificación de cada uno de los modelos de MASINA, incluso los de inteligencia colectiva e individual). De, todos estos diagramas, se tomarán en cuenta para la generación del código el modelo de agentes y modelo de tareas ya que estos proveen la información necesaria para generar la estructura inicial de las clases que determinarán los agentes (la estructura de los agentes se puede generar solo usando estos dos diagramas, el funcionamiento en sí del agente debe ser codificado por el usuario y queda para futuras investigaciones), por su parte lo referido a los procesos de inteligencia y aprendizaje deben ser desarrollados manualmente por el usuario debido a que son procesos muy complejos y queda a disposición personal de dicho usuario definir cómo implementarlos. Es así, como el modelo de comunicaciones, coordinación, inteligencia individual y colectiva se utilizarán para generar documentación y comentarios en los archivos generados para el proyecto. En efecto, V-MAS genera una clase de Java para cada agente en la que se integran los objetos de tipo Servicio que hayan sido definidos por el modelador, así como las tareas correspondientes a cada servicio. A su vez, debido a que la plataforma destino se basa en el lenguaje Java, se toman en cuenta factores como sintaxis del lenguaje, específicamente la estructura de clases aceptadas en la plataforma JADE, tipo de datos soportados, herencia, modificadores de acceso, así como métodos heredados de clases abstractas. A su vez, el PSM está representado por cada una de las clases generadas por VMAS, para cada agente registrado se generará un archivo con extensión *.java* que contendrá la clase principal del agente, las clases relacionadas a cada uno de sus servicios, así como los métodos y/o clases que representan las tareas que lleva a cabo cada servicio del agente. De igual forma, el archivo de agente generado contendrá clases auxiliares que servirán como artificio para almacenar los datos de retorno de cada uno de los servicios y tareas registrados.

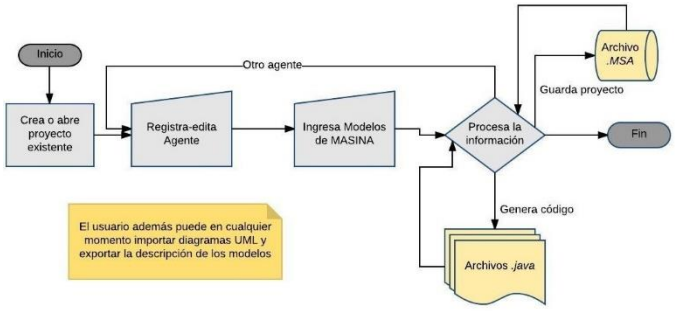

Figura 2: Diagrama de Flujo de las Actividades Desarrolladas en la Aplicación

En la Figura 2 se describen las actividades que los usuarios pueden desarrollar en V-MAS a través de un diagrama de flujo (en ella se especifica el ciclo de vida de un proyecto). El usuario comienza creando un proyecto, crea los agentes del SMA especifica la información de los modelos para cada uno de ellos, posteriormente puede generar el código Java y/o guardar el proyecto en un archivo con extensión MSA.

La clase principal del agente generado por V-MAS se define según se muestra en la Figura 3. En ella se declaran y se registran cada uno de los servicios del agente en el Directorio Facilitador (DF) de JADE, de acuerdo a lo que el usuario especifique en el modelo de agentes ingresado en V-MAS; además, como parte de las especificaciones de JADE, la clase Agente debe heredar de la clase *Agent* y contener el método *setup* que se ejecutará al agregar el agente al contenedor (este método registra los servicios del agente en el DF de JADE) y el método *takeDown* que se ejecuta cuando el agente se libera, a fin de realizar tareas de limpieza, como es el caso de eliminar los servicios del agente que fueron registrados en el DF.

Vale destacar, que cada servicio del agente representa un Comportamiento (Behaviour) de JADE, para el cual se genera una clase interna. Asimismo, las tareas registradas en el modelo de tareas, representaran una nueva clase interna que hereda de un comportamiento o un método de la clase, de acuerdo a como lo haya definido el usuario a la hora de ingresar la información.

```
public class Mi_Agente extends Agent{
    protected void setup(){
        DFAgentDescription dfd =  new DFAgentDescription();
        dfd.setName(getAID());

        ServiceDescription miServicio = new ServiceDescription();
        miServicio.setType("ServiceMiServicio");
        miServicio.setName(getLocalName());
        addBehaviour( new MiServicio(myAgent));
        try {
            DFService.register(this, dfd);
        }catch (Exception e){}
    }

    protected void takeDown(){
        try { DFService.deregister(this); }
        catch (Exception e){}
    }
}
```

Figura 3: Código Generado para la Clase Agente

```
public class Mi_Servicio extends Behaviour{
    private int calidad = 0;
    private MiServicioReturnValue valorRetorno;

    public void action(){...3 lines }

    public int getCalidad(){...3 lines }

    public void setCalidad(int calidad){...3 lines }

    public void miTarea1 (String ingrediente1, String ingrediente2){.

    public class miTarea2 extends Behaviour {...9 lines }

    public boolean done() {...3 lines }

    public class MiServicioReturnValue {...14 lines }
}
```

Figura 4: Código Generado para los Servicios

Por otro lado, la estructura básica de las clases que representan los servicios se define en la Figura 4, es una clase con muchos elementos que contiene los métodos relacionados a las tareas asociadas, un método *action*, atributos del servicio. Esta clase

debe heredar del comportamiento JADE adecuado, según lo que haya indicado el usuario al momento de especificar el servicio.

Seguidamente, el usuario puede registrar por cada agente uno o más servicios. De cada servicio se crea una clase cuyo nombre se corresponde con el nombre del servicio, y cuyos métodos y propiedades dependerán de las propiedades registradas y del tipo de comportamiento definido por el usuario. Además, los parámetros de salida se plantean como una serie de variables que se agrupan en un objeto de una clase auxiliar denominado *ValorRetorno* (ver Figura 5), el nombre de éstas clases se genera a través de la siguiente fórmula: *<nombre del servicio> + < "ReturnValue">*.

Finalmente, se deben agregar los métodos o clases correspondientes a cada una de las tareas asociadas al servicio, ésta información se obtiene del modelo de tareas del agente en el cual se especifica nombre de la tarea, ingredientes (parámetros), el servicio al que pertenece, y opcionalmente, el tipo de comportamiento que ésta posee. En caso que la tarea se defina como un comportamiento, se creará una clase de Java que hereda del comportamiento indicado, de otra manera la tarea pasará a ser un método de la clase del servicio al cual está ligada. En la Figura 5 se define la estructura básica de las clases auxiliares, ésta información se extrae de los parámetros de salida especificados para cada servicio, es decir, para la clase a generar se definirá una serie de atributos consistentes con los parámetros de salida del servicio y según el tipo de dato ingresado, además de establecerle los modificadores de acceso, y los respectivos métodos *getters* y *setters*.

```
public class MiServicioReturnValue {
    char miParametro1;

    public MiServicioReturnValue() {
    }

    public char getMiParametro1() {
        return miParametro1;
    }

    public void setMiParametro1(char miParametro1) {
        this.miParametro1 = miParametro1;
    }
}
```

Figura 5: Código Generado para las Clases Auxiliares

Esencialmente, la evolución del archivo de salida para cada uno de los agentes va cambiando según la información que se introdujo en los modelos, así como de la configuración de generación de código establecida por el usuario. En la Figura 6 se describe el aporte que realiza cada modelo sobre el archivo generado.

Finalmente, pero no menos importante, la aplicación ofrece una serie de funcionalidades que ayudarán al usuario en el alcance de ciertas actividades que son de uso común en las herramientas CASE, entre ellas la exportación de los modelos de MASINA de forma tabular para cualquier procesador de texto, de tal forma que se apoye al desarrollador en la creación de documentos académicos, documentación del proyecto, artículos científicos, etc., esto se logra haciendo uso del portapapeles del sistema operativo. A su vez, cuenta con un módulo de administración de proyecto en el cual se incluyen varias funcionalidades, ellas son: guardar proyecto, guardar como y abrir proyecto, un módulo de configuración de parámetros y la posibilidad de importar la información UML correspondiente a cada uno de los agentes modelados.

Las tecnologías involucradas en el desarrollo de Visual MAS Modeller son diversas, principalmente el desarrollo se rigió bajo la plataforma Java versión 8 e interactúa con archivos de entrada/salida XML y UML. Por otro lado, SQLite se utiliza como base de datos de la aplicación y patrones de diseño para las tareas y procesos que la herramienta deberá ejecutar. Se seleccionó JADE como plataforma objetivo, es decir, la plataforma para la cual sería generado el código debido a que es ampliamente utilizada y existe abundante documentación en la web. Por su parte, JADE [4] define la plataforma como: es un software totalmente implementado en lenguaje Java. Simplifica la implementación de sistemas multi-agentes a través de un middleware que cumpla con las especificaciones FIPA y a través de un conjunto de herramientas que soportan las fases de depuración y despliegue.

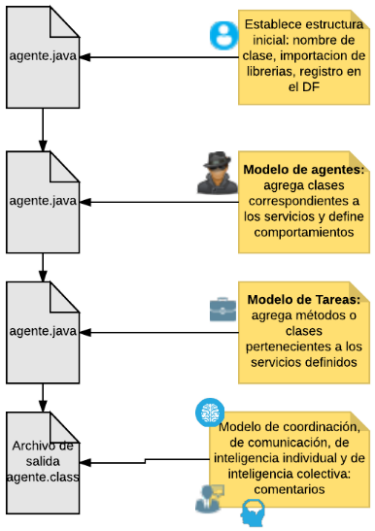

Figura 6: Aporte de cada Modelo sobre el Código Generado

V. Caso de Estudio

Con el fin de exponer el uso de la aplicación en un ambiente real, se propone un caso de estudio basado en el juego de TicTacToe, éste consiste en 2 tipos de agentes, un agente que representa al jugador y otro para representar el tablero del juego. Éstos agentes interactúan entre sí para informar la jugada seleccionada por parte de los jugadores; mientras que el agente tablero informará acerca del estado actual de la partida. A fin de abreviar la información, se especificará solo los modelos de MASINA que son cruciales para la generación del código, siendo ellos las secciones objetivos y servicios, modelo de agentes y el modelo de tareas.

A. *Agente Jugador*

Agente que representa una entidad de tipo jugador, se encarga de evaluar las jugadas e informar acerca del próximo movimiento. La descripción del objetivo del modelo de

agentes para esta entidad se cargó en la aplicación, según se observa en la Figura 7.

Por otro lado, se definió un servicio encargado principalmente de solicitar jugadas al tablero, según se describe en la Figura 8.

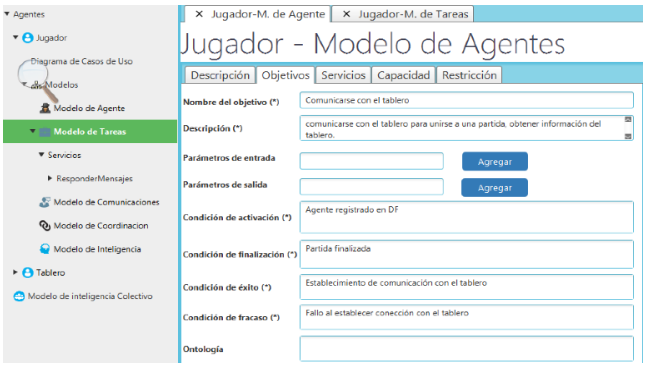

Figura 7: Definición del Objetivo Comunicarse con el Tablero

Además, a este servicio se le definió una tarea denominada "*Preparar Respuesta"*, como un método encargado de gestionar los mensajes de entrada y salida cuya descripción se detalla en la Figura 9. De igual forma el servicio cuenta con otra tarea que se representa como un comportamiento que extiende de *Behaviour*. Esta tarea se denomina "*Enviar Movimiento*" y representa el proceso responsable de llevar la secuencia de la partida y enviar el siguiente movimiento del jugador definido en la Figura 10.

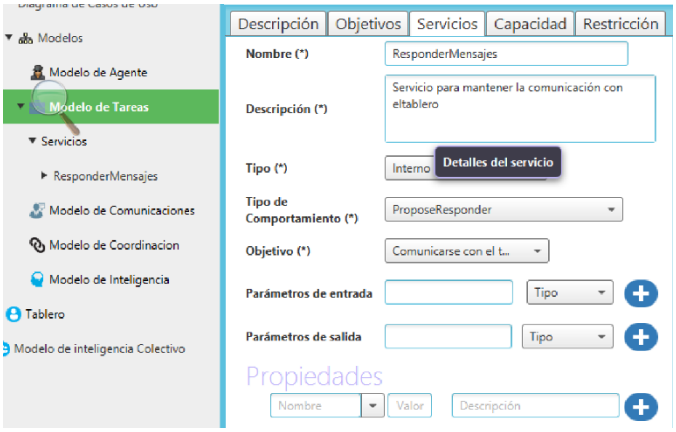

Figura 8: Definición del Servicio Solicitar Jugada

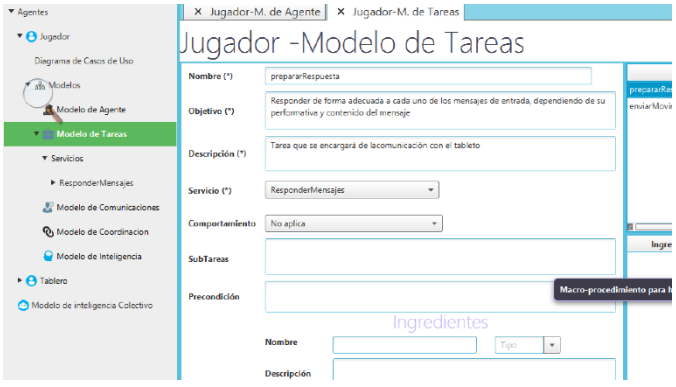

Figura 9: Definición de la Tarea Preparar Respuesta

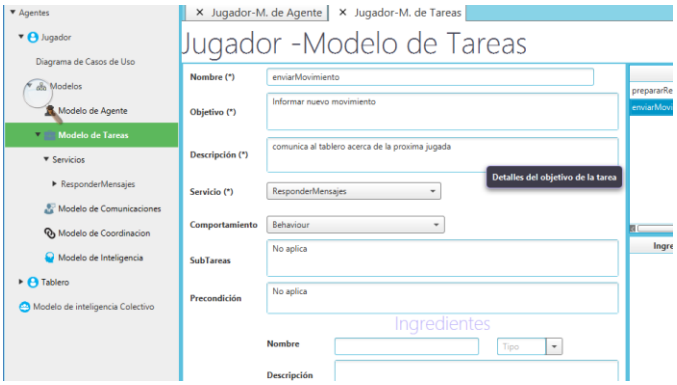

Figura 10: Definición de la Tarea Enviar Movimiento

B. Agente Tablero

Este agente crea y gestiona cada una de las partidas que se dan en la aplicación y coordina los mensajes que envían cada uno de los jugadores para informar acerca de su próximo movimiento en el tablero. Su objetivo es encargarse de gestionar las partidas, es decir, iniciarlas, coordinar movimiento de los jugadores, decidir ganador y mostrar resultados. La Figura 11 muestra la información cargada para este agente en V-MAS.

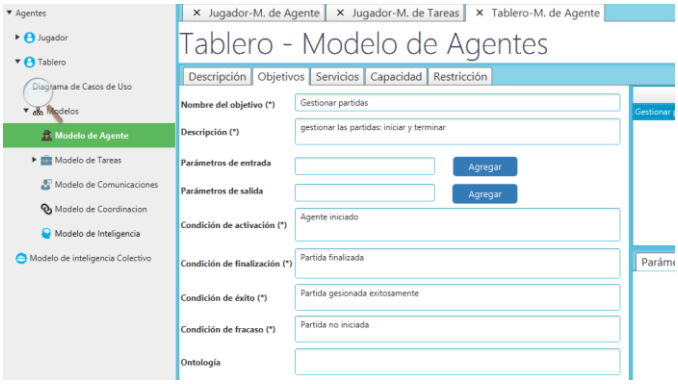

Figura 11: Definición del Objetivo Gestionar Partidas

Para este agente, se registraron 3 servicios en V-MAS Modeller, ellos son: *ComenzarPartida* cuya descripción se muestra en la Figura 12, *CrearPartida* definido según la Figura 13 y finalmente *Mover* que se describe en Figura 14.

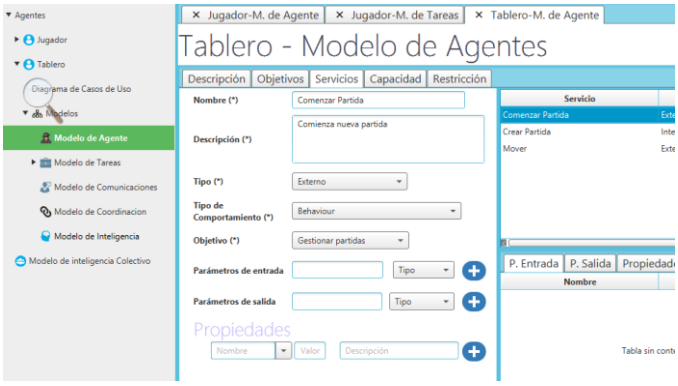

Figura 12: Definición del Servicio Comenzar Partida

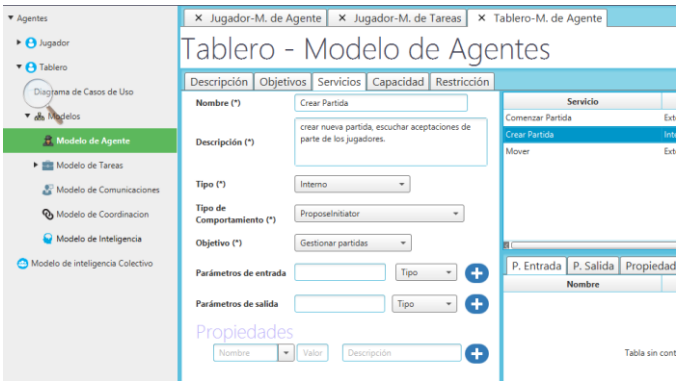

Figura 13: Definición del Servicio Crear Partida

Seguidamente, se definió una serie de tareas asociadas a los servicios, a continuación, se detalla con más profundidad cada una de ellas:

Aceptar Propuesta: esta tarea hace parte del servicio *Crear Partida* y se encargará de realizar las acciones correspondientes para iniciar una partida cuando un jugador acepte unirse a ella. La descripción de ésta tarea se define a continuación en la Figura 15.

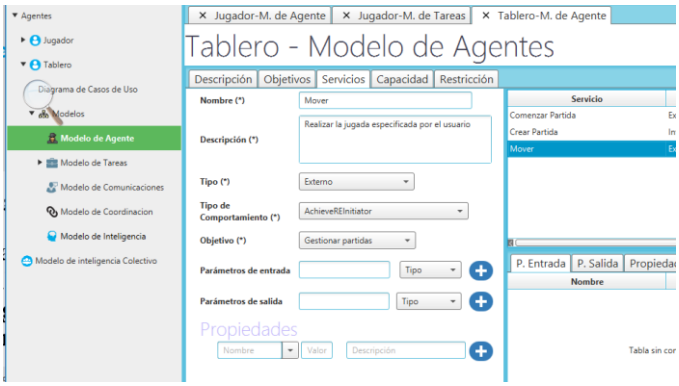

Figura 14: Definición del Servicio Mover

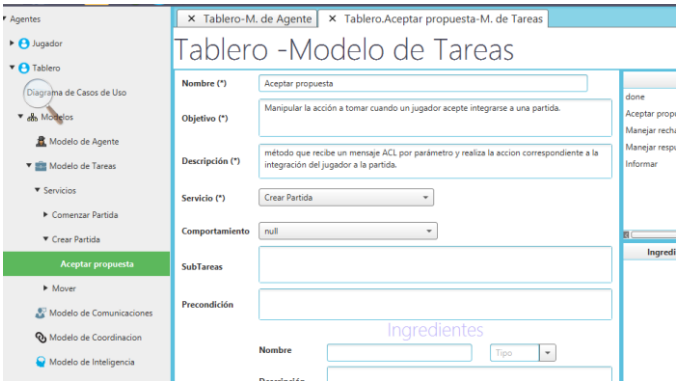

Figura 15: Definición de la Tarea Aceptar Propuesta

Manejar rechazo: Esta tarea hace parte del servicio *Mover* y se encarga principalmente de realizar las acciones necesarias, cuando un jugador abandona la partida, la descripción de ésta tarea se muestra en la Figura 16.

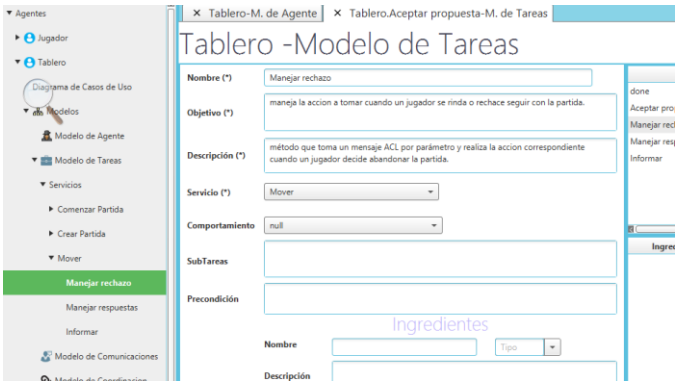

Figura 16: Definición de la Tarea Manejar Rechazo

Manejar respuestas: Esta tarea hace parte del servicio *Mover*, su función primordial es escuchar mensajes de parte de los jugadores y tomar las acciones requeridas según el contenido de los mismos para efectuar los movimientos en el tablero. La Figura 17 describe con detalle la definición de esta tarea, siguiendo el modelo MASINA.

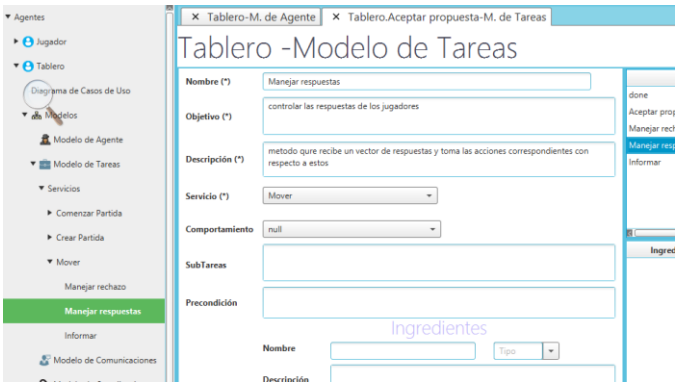

Figura 17: Definición de la Tarea Manejar Respuestas

Informar: Esta tarea hace parte del servicio *Mover*, principalmente se encarga de difundir el estado actual de la partida: posiciones, condición del tablero y ganador. En la Figura 18 se especifica la descripción de ésta tarea siguiendo el modelo MASINA.

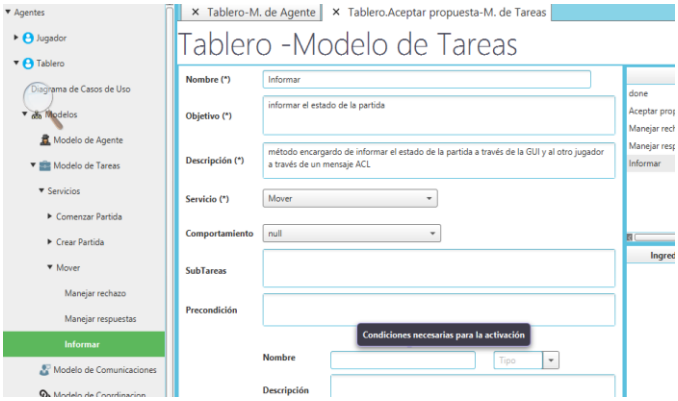

Figura 18: Definición de la Tarea Informar

Preparar Respuesta: tarea que hace parte del servicio *Responder Mensajes* y su función principal es responder a cada uno de los mensajes de entrada de manera correcta. En la Tabla

I se detalla las propiedades de esta tarea, siguiendo la plantilla propuesta por MASINA para el modelo de tareas, ésta tabla se exportó directamente desde la aplicación haciendo uso de la funcionalidad *copiar modelo*.

Tabla I: Descripción de la Tarea Preparar Respuesta

Modelo de Tareas	
Nombre	Preparar Respuesta
Objetivo	Responder de forma adecuada a cada uno de los mensajes de entrada, dependiendo de su performativa y contenido del mensaje
Descripción	Tarea que se encargará de la comunicación con el tablero
Servicios asociados	ResponderMensajes
Subtareas	No aplica
Ingredientes	No aplica

Enviar Movimiento: ésta tarea hace parte del servicio *Responder Mensajes* y su función principal es reportar al tablero acerca de la jugada que el participante desea realizar. En la Tabla II se detalla esta tarea siguiendo la plantilla propuesta por MASINA para el modelo de tareas, ésta tabla se exportó directamente desde la aplicación haciendo uso de la funcionalidad *copiar modelo*.

Tabla II: Descripción de la Tarea Enviar Movimiento

Modelo de Tareas	
Nombre	Enviar Movimiento
Objetivo	Informar nuevo movimiento
Descripción	comunica al tablero acerca de la próxima jugada
Servicios asociados	Responder Mensajes
Subtareas	No aplica
Ingredientes	No aplica

Seguidamente, se procedió a generar el código base para cada uno de los agentes registrados (la Figura 19 muestra como el usuario puede generar el código utilizando V-MAS), para los cuales V-MAS Modeller creó un archivo con extensión *.java* para cada agente, especificando elementos que constituyen la estructura de la clase del agente, constructores y clases asociadas a los servicios, entre otros. La configuración seleccionada (ver Figura 19) para la generación de código constó en no agregar comentarios y suprimir los espacios en blanco para aquellos elementos que definen un nombre de clase, atributo o método.

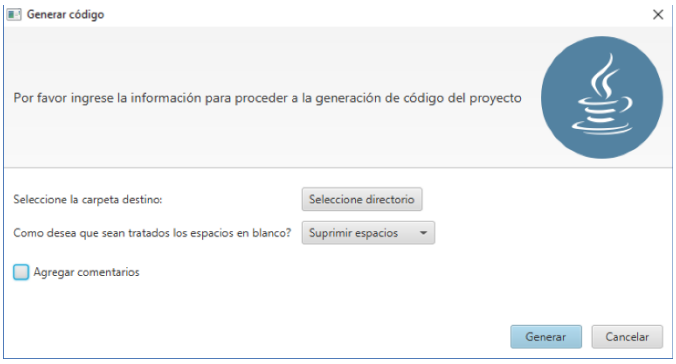

Figura 19: Interfaz para Configurar la Generación del Código

La Figura 20 muestra parte del código generado por V-MAS para el agente Jugador, mientras que en la Figura 21 se muestra un segmento de código correspondiente al agente Tablero, donde se detallan los métodos setup y takeDown los cuales han sido generados completamente por V-MAS, y no requieren de mucha configuración adicional por parte del usuario.

Una vez obtenidas estas clases desde V-MAS se procedió a desarrollar el código para la implementación de la lógica de cada uno de los agentes, seguidamente, se compiló el código en Java y se ejecutó la aplicación en la plataforma JADE. Posteriormente se desplegaron 2 agentes de tipo jugador denominados jugador1 y jugador2, y uno de tipo Tablero denominado tablero.

En el mismo sentido, en la Figura 22 se muestra una imagen de los agentes que se encuentran desplegados en la plataforma JADE.

```
public class Jugador extends Agent {

    protected void setup() {

        DFAgentDescription dfd = new DFAgentDescription();
        dfd.setName(getAID());

        ServiceDescription respondermensajes = new ServiceDescription();
        respondermensajes.setType("respondermensajes");
        respondermensajes.setName(getLocalName());
        addBehaviour(new Respondermensajes(this, mt));
        try {
            DFService.register(this, dfd);
        } catch (Exception e) {
        }

    }

    //*Fin de metodo*//

    protected void takeDown() {
        try {
            DFService.deregister(this);
        } catch (Exception e) {
        }
    }
```

Figura 20: Sección del Código Generado Automáticamente para el Agente Jugador

```
public class Tablero extends Agent {

    protected void setup() {

        DFAgentDescription dfd = new DFAgentDescription();
        dfd.setName(getAID());

        ServiceDescription comenzarPartida = new ServiceDescription();
        comenzarPartida.setType("comenzarPartida");
        comenzarPartida.setName(getLocalName());
        addBehaviour(new ComenzarPartida(this));

        ServiceDescription crearPartida = new ServiceDescription();
        crearPartida.setType("crearPartida");
        crearPartida.setName(getLocalName());
        addBehaviour(new CrearPartida(this, msg));

        ServiceDescription mover = new ServiceDescription();
        mover.setType("mover");
        mover.setName(getLocalName());
        addBehaviour(new Mover(this, msg));
        try {
            DFService.register(this, dfd);
        } catch (Exception e) {
        }

    }
```

Figura 21: Sección del Código Generado Automáticamente para el Agente Tablero

Por su parte, en la Figura 23 se muestra la interacción que se da entre los agentes Jugador y Tablero haciendo uso de la herramienta *sniffer* que provee la plataforma JADE. Allí se observa como fluyen los mensajes entre los agentes del sistema mientras se desarrolla el juego.

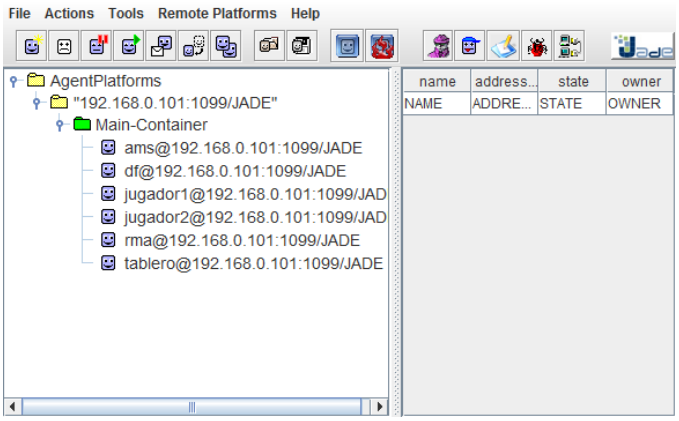

Figura 22: Contenedor de JADE con los Agentes en Ejecución

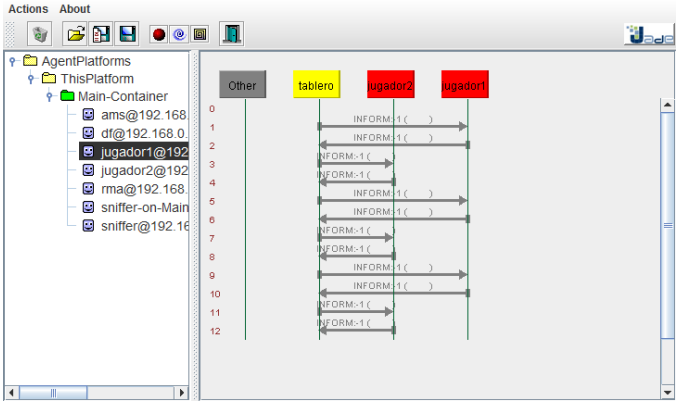

Figura 23: Visualización de la Interacción entre los Agentes a través del Sniffer de JADE

VI. Experimento

Para realizar una validación más científica de la herramienta, se tomó una sección de Inteligencia Artificial de la Universidad Nacional Experimental del Táchira de 57 estudiantes y se dividió en clases de la siguiente forma:

- No conocen MASINA.
- Nivel medio en el uso de MASINA.

Cada clase se dividió en grupos de tres estudiantes y se les instruyó para que crearan un SMA simple de solo dos agentes (el clásico ejemplo productor-consumidor o vendedor-comprador). Luego se crearon sub clases de la siguiente manera:

- No conocen MASINA.
  - 5 grupos deben hacer todo manualmente.
  - 5 grupos deben usar V-MAS
- Nivel medio en el uso de MASINA.
  - 4 grupos deben hacer todo manualmente.
  - 4 grupos deben usar V-MAS.

A cada grupo se les tomo el tiempo en horas que duraron para desarrollar el SMA (sólo debían crear el modelo de agentes, el modelo de tareas y codificar), a fin de poder comparar si disminuía usando MASINA o no. Todos los grupos tenían conocimiento medios de JADE y se les indicó en forma general las tareas y servicios con la que debía contar cada agente, de tal forma que esto no influyera en el tiempo de desarrollo.

Las métricas a tomar en cuenta son el tiempo de desarrollo, él número de errores y la necesidad de un experto para poder completar los modelos.

De los grupos que no conocían MASINA, y debían hacer todo manualmente, ninguno pudo crear los modelos sin ayuda del profesor, mientras que de los que usaron V-MAS sólo uno necesitó ayuda del profesor para ingresar los modelos. Aun así, los que recibieron ayuda del profesor y trabajaron manualmente requirieron mayor tiempo para finalizar la fase de diseño y sus modelos contaban con errores, en algunos parámetros.

Por su parte de los grupos que conocían MASINA a nivel medio, ninguno requirió ayuda del profesor, pero los que lo hicieron manualmente terminaron la fase de diseño con unos pocos minutos de retraso. De igual forma sus modelos contenían algunos errores, en los parámetros.

El profesor instruyó a los estudiantes para que corrigieran los errores, con lo cual perdieron aún más tiempo.

Seguidamente se pasó a la fase de desarrollo; los estudiantes que trabajaron con V-MAS tuvieron el código inicial del proyecto libre de errores en apenas unos segundos, mientras que los que hicieron el sistema manualmente, tardaron entre 20 y 30 minutos en completar el código inicial del proyecto; en ese tiempo, los que usaron V-MAS ya habían terminado de codificar y probar todas las funcionalidades del sistema.

VII. Conclusiones y Trabajo Futuro

La presente investigación concluyó en el desarrollo de una aplicación CASE que permite a los diseñadores de sistemas multi-agentes modelar aplicaciones siguiendo la arquitectura o

enfoque de SMA, a su vez ofrece a los usuarios la posibilidad de exportar los modelos a través de las plantillas propuestas por MASINA y generar el código base en JADE para cada uno de los agentes registrados en el proyecto. Esta aplicación permite a los desarrolladores y/o diseñadores crear, guardar, editar y visualizar un proyecto, además permite crear Sistemas multi-agentes siguiendo una metodología apropiada para tal fin, así como facilitar al usuario la creación de los modelos de la metodología en sí. Por otro lado, la aplicación desarrollada ayuda a ahorrar tiempo de desarrollo ya que se automatizan ciertas actividades que normalmente se desarrollan en forma manual.

Visual MAS Modeller se diferencia principalmente de las herramientas previamente desarrolladas (SISGECOMA y EDISMA) en que ellas generan código para plataformas poco conocidas y usadas, su código no se rige bajo el estándar FIPA, mientras que V-MAS Modeller genera el código base para una plataforma de agentes ampliamente usada y conocida como lo es JADE.

Por otro lado, V-MAS realiza a lo largo del proceso de introducción de datos un conjunto de validaciones que permiten al usuario la obtención de un código válido y libre de errores de sintaxis, además, que éste cumple con los estándares de definición de clases, atributos y métodos establecidos en Java. A su vez, la aplicación fue desarrollada pensando en el usuario, para facilitar el proceso de introducción de información, proveyendo una interfaz intuitiva, clara y efectiva para hacer este proceso bastante sencillo. Aunado a esto, V-MAS Modeller apoya al diseñador del SMA en la generación de documentación o informes ofreciéndole la posibilidad de copiar los modelos de forma tabular que coinciden con las plantillas establecidas en MASINA. A continuación, en la Tabla III se establece una comparación entre las herramientas que han sido desarrolladas previamente y Visual MAS Modeller.

De la Tabla III, se observa que herramientas similaresa V-MAS, no generan código para plataformas de gestión de agentes, como es el caso de C/C++, lo que le impone al desarrollador del sistema el trabajo de tener que codificar todos los elementos que le permitan al sistema multi agentes co-existir (registrar, descubrir, comunicar, interactuar, etre otros), lo cual genera aún más trabajo para el usuario. De la misma forma, la información tomada por otras herramientas no cuenta con validación de datos, lo que permite que el usuario pueda saltarse pasos importantes en la metodología, o ingresar paramétros que no se corresponde con los valores requeridos por la metodología, lo cual inducirá a errores de sintaxis en el código generado. De igual forma, la usabilidad de V-MAS es superior a la de las demás herramientas, ya que cuenta con datos parametrizados que son mostrados en listas y otros componentes gráficos y a su vez muestra ayudas que le indican al usuario que iformación ingresar en cada campo, algo de lo que carecen las demás herramientas.

Por otra parte, aunque V-MAS solo genera código para la plataforma JADE, su arquitectura se ha desarrollado de manera flexible usando tecnicas de desarrollo de software orientado a objetos y patrones de diseño que permitiran incorporar en el futuro nuevas plataformas de agentes como código objetivo de V-MAS.

Tabla III: Comparación de las Herramientas

Aplicación	**Características**
V-MAS Modeller	**Salida:** archivos de clase .java para cada agente **Metodología:** MASINA **Plataforma Destino:** JADE **Validación de datos**: Sí **Exportación de modelos:** Sí **Importación de información UML:** StarUML y ArgoUML **Usabilidad:** Alta
EDISMA	**Salida:** Archivos de clases en C++. **Metodología:** MASINA **Plataforma Destino:** C++ **Validación de información:** Deficiente **Exportación de modelos:** No soportado **Importación de información UML:** Umbrello **Usabilidad:** Media
SISGECOMA	**Salida:** Archivos compilados en C **Metodología:** MASINA **Plataforma Destino:** C **Validación de información:** Deficiente **Exportación de modelos:** No soportado **Importación de información UML:** No soportado **Usabilidad:** Media

El experimento realizado con los estudiantes logró demostrar que V-MAS no solo ayuda a reducir el tiempo de desarrollo sino que facilita el entendimiento a la hora de diseñar el sistema multi-agentes. Lo que permitió validar que V-MAS cumple con los objetivos de diseño.

Finalmente, con el fin de lograr que V-MAS sea un software que responda correctamente ante las diversas situaciones que se pueden presentar en el modelado de SMA se presentó un caso de estudio donde se diseñaron y se ejecutaron una serie de pruebas que permiten verificar el funcionamiento del software,tomando en consideración conceptos de calidad de software, como lo son valores al límite, criterio de clases válidas y escenarios.

El trabajo futuro, se enfoca principalmente en generar código para otras plataformas de agentes conocidas, así como la utilización de otros modelos de MASINA para ampliar aún más el código generado.

REFERENCIAS

[1] M. Maalal and A. Addou. *A New Approach of Designing Multi-Agent Systems*. International Journal of Advanced Computer Science and Applications, Vol. 2, No. 11. Casablanca, Marruecos, 2011.

[2] J. Aguilar, I. Bessembel, M. Cerrada, F. Hidrobo y F. Narciso. *Una Metodología para el Modelado de Sistemas de Ingeniería Orientado a Agentes*. Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial, Vol. 12, No. 38, pp. 39-60 Asociación Española para la Inteligencia Artificial. España, 2008.

[3] R. Mall. *Fundamentals of Software Engineering*. Prentice Hall of India, New Delhi, 2003.

[4] JADE. http://jade.tilab.com.

[5] G. Weiss. *Multiagents Systems 1st ed*. Cambridge MA. MIT Press, 2013.

[6] C. Iglesias, M. Garijo, J. González, and J. Velasco. *Analysis and Design of Multiagent Systems using MAS-CommonKADS*. Intelligent Agents IV, 1365, pp. 313-327, 1998.

[7] B. Burmeister. *Models and Methodology for Agent-Oriented Analysis and Design*. In Working Notes of the KI'96 Workshop on Agent-Oriented Programming and Distributed Systems, Eindhoven, The Netherlands, 1996.

[8] S. A. DeLoach. *Modeling Organizational Rules in the Multi-agent Systems Engineering Methodology*. In Canadian Conference on AI, pp. 1–15, 2002.

[9] J. DiLeo, T. Jacobs, and S. A. DeLoach. *Integrating Ontologies into Multiagent Systems Engineering*. In AOIS@CAiSE, 2002.

[10] C. A. Iglesias, M. Garijo, and J. Centeno- González. *A Survey of Agent-Oriented Methodologies.* In ATAL 98: Proceedings of the 5th International Workshop on Intelligent Agents V, Agent Theories, Architectures, and Languages, pp. 317–330, London, UK, 1999.

[11] D. Kinny, M. Georgeff, and A. Rao. *A Methodology and Modelling Technique for Systems of BDI Agents*. In Seventh European Workshop on Modelling Autonomous Agents in a Multi-Agent World, Eindhoven, The Netherlands, 1996.

[12] A. L. Self and S. A. DeLoach. *Designing and Specifying Mobility within the Multiagent Systems Engineering Methodology*. In SAC'03: Proceedings of the 2003 ACM Symposium on Applied Computing, pp. 50–55, New York, NY, USA. ACM Press, 2003.

[13] S. Vafadar, A. Barfouroush, and M. Ayatollahzadeh. *Towards a More Expressive and Refinable Multiagent System Engineering Methodology*. In AOIS, pp. 126–141, 2003.

[14] M. F. Wood and S. A. DeLoach. *An Overview of the Multiagent Systems Engineering Methodology.* In First International Workshop on Agent-Oriented Software Engineering (AOSE 2000), pp. 207–221, Secaucus, NJ, USA. Springer-Verlag New York, Inc, 2001.

[15] F. Alonso, S. Frutos, L. Martinez, and C. Montes. *SONIA: A Methodology for Natural Agent Development.* In Fifth International Workshop Engineering Societies in the Agents World, Toulouse, France, October 2004.

[16] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos. *Tropos: An Agent-Oriented Software Development Methodology*. Autonomous Agents and Multi-Agent Systems, 2004.

[17] A. Collinot, P. Carle, and K. Zeghal. *Cassiopeia: A Method for Designing Computational Organizations.* In The First International Workshop: Decentralized Multi-Agent Systems DIMAS'95, pp. 124–131, Crakow, Poland, November 1995.

[18] M. Elammari and W. Lalonde. *An Agent-Oriented Methodology: High-level and Intermediate Models.* In the First International Bi-Conference Workshop on Agent-Oriented Information Systems, Seattle, USA and Heidelberg, Germany, May-June 1999.

[19] A. Omicini. *SODA: Societies and Infrastructures in the Analysis and Design of Agent-Based Systems.* In Agent-Oriented Software Engineering, pp. 185–194. Springer, LNCS 1957, 2001.

[20] L. Padgham and M. Wimikoff. *Prometheus: A Methodology for Developing Intelligent Agents*. In Agent-Oriented Software Engineering III, pp. 174–185. Springer, LNCS 2585, 2003.

[21] M. Wooldridge, N. R. Jennings, and D. Kinny. *The Gaia Methodology for Agent-Oriented Analysis and Design.* Autonomous Agents and Multi-Agent Systems, 2000.

[22] F. Zambonelli, N. R. Jennings, and M. Wooldridge. *Developing Multiagent Systems: The Gaia Methodology.* ACM Transaction on Software Engineering and Methodology, 2003.

[23] R. Bravo. *SISGECOMA. Sistema Generador de Código para la Metodología MASINA.* Universidad de Los Andes, Venezuela. 2009.

[24] FIPA.http://www.fipa.org.

[25] J. Whitten and L. Bentley. *Análisis y Diseño de Sistemas de Información*. México: McGraw-Hill. 2003.

[26] P. Rivero. *EDISMA: Entorno de Desarrollo Integrado para la Creación de Sistemas MutiAgentes*. Universidad de Los Andes, Venezuela. 2013.

[27] A. Mészáros. *Code Generation from AML to Jadex*. Comenius University in Bratislava. Eslovaquia. 2010.

[28] J. Aguilar y R. Castellanos. *Modelo Ontológico de Verificación de Sistemas Multiagentes Diseñados bajo MASINA*. Avances en Sistemas e Informática, September 1997.

[29] S. Goyal. Major *Seminar on Feature Driven Development.* Munich. Technical University Munich. 2007.

[30] L. Calabria. *Metodología FDD*. Universidad ORT Uruguay. Facultad de Ingeniería. 2003.

Índice de Autores

REVECOM

Sociedad Venezolana de Computación

La Sociedad Venezolana de Computación está comprometida con el impulso de una nueva generación académica y profesional en nuestra área de saber para el desarrollo del país.

Los conceptos y puntos de vista expresados en los trabajos publicados en este libro representan las opiniones personales de los autores y no reflejan el juicio de los editores o de la Sociedad Venezolana de Computación.

ISSN: 2244-7040

9 772244 704006

www.svc.net.ve/revecom