Módulo de generación de aplicaciones multi-dispositivo a partir del procesamiento de imágenes

Laura Sánchez Morales , Viviana Yarel Rosales Morales, Giner Alor Hernández, Rubén Posada Gómez, Hilarión Muñoz Contreras, Ulises Juárez Martínez

División de Estudios de Posgrado e Investigación, Instituto Tecnológico de Orizaba, Orizaba, Veracruz, México

lauransanchezmorales@gmail.com, viviana_rosales@outlook.com, galor@itorizaba.edu.mx, rposada@itorizaba.edu.mx, hmunoz@itorizaba.edu.mx, ujuarez@itorizaba.edu.mx

Resumen. En la actualidad existen diferentes esquemas de generación automática de software como son MDA (*Model Driven Architecture*), FDD (*Feature-driven developmen*), RAD (*Rapid Application Development*), por mencionar algunos. Sin embargo hasta el momento no se ha abordado la generación de software mediante el uso de técnicas de inteligencia artificial como son el procesamiento digital de imágenes y reconocimiento de patrones. En este trabajo se propone el desarrollo de un componente de software que utilice técnicas de procesamiento de imágenes y reconocimiento de patrones para la generación automática de aplicaciones multi-dispositivo. Para lograr este objetivo, se identifican los elementos de una imagen y se genera el código de una aplicación multi-dispositivo con los elementos detectados previamente seleccionados en dicha imagen. Finalmente se presenta un caso de estudio que permite describir la funcionalidad del módulo propuesto.

Palabras clave: aplicaciones multi-dispositivo, generación automática de software, procesamiento de imágenes.

1. Introducción

El desarrollo de software está en constante evolución, y a través del tiempo se propusieron diversos enfoques de desarrollo como FDD (*Feature-driven development*), MDA (*Model Driven Architecture*), RAD (*Rapid Application Development*), por mencionar algunos; con la finalidad de agilizar el tiempo de desarrollo sin comprometer la calidad del producto final. Sin embargo, la mayoría de las herramientas empleadas por desarrolladores, y todos aquellos interesados en el desarrollo de software, ofrecen características de desarrollo muy similares entre sí pero son pocas las que, con la finalidad de satisfacer estas necesidades implementan técnicas de inteligencia artificial como el reconocimiento de patrones en imágenes. Desde esta perspectiva surge uno de los principales motivos para realizar el presente trabajo.

Por otra parte las aplicaciones multi-dispositivo como su nombre lo indica, permiten el desarrollo en un lenguaje o conjunto de lenguajes y su posterior ejecución en diferentes dispositivos de hardware. Sin embargo esto no implica que se hayan cubierto

todas las necesidades para la generación automática de código, por el contrario, cada vez surgen más alternativas para la generación automática de software. El procesamiento digital de imágenes es el conjunto de técnicas que se aplican a las imágenes digitales con el objetivo de mejorar su calidad o facilitar la búsqueda de información inmersa en ellas. En este sentido el Tratamiento Digital de Imágenes contempla el procesamiento y el análisis de imágenes. Este procesamiento se refiere a la realización de transformaciones y a la restauración y mejoramiento de las imágenes. El análisis consiste en la extracción de propiedades y características de las imágenes, así como la clasificación e identificación y el reconocimiento de patrones [1]. De ahí que la importancia del procesamiento y análisis de imágenes digitales esté presente en diversas áreas, tales como la medicina [2], biología [3], astronomía [4], fotografía [5], historia [6] y geología [7], entre otras. Esto se debe a que la obtención de imágenes no está limitada por el dominio de aplicación. Sin embargo en la ingeniería de software hasta el momento no se reportan métodos o técnicas ampliamente definidos sobre el tratamiento de imágenes para el desarrollo automático de software. Con base en lo anterior este trabajo propone el desarrollo de un componente de software que identifique elementos en una imagen mediante técnicas de procesamiento de imágenes y genere el código de una aplicación multi-dispositivo con los elementos identificados en la imagen.

Este documento está estructurado de la siguiente manera, en la sección 2 se presenta el estado del arte referente a los diversos trabajos relacionados para la generación automática de software. En la sección 3 se describe el proceso planteado para la generación de software. En la sección 4 se presentan parte de las especificaciones definidas para la identificación de elementos en interfaces. En la sección 5 se presenta un caso de estudio empleando una imagen que representa la portada principal de una página Web. Finalmente en la sección 6 se presentan las conclusiones y trabajo a futuro.

2. Estado del arte

A continuación se presenta la revisión del estado del arte sobre los trabajos relevantes en el área de la generación automática de código para el desarrollo de software, las cuales se encuentran clasificadas por herramientas para desarrollo o generación de aplicaciones y técnicas de reconocimiento y procesamiento de imágenes.

2.1. Herramientas para desarrollo o generación de aplicaciones

En [8] se presentó una herramienta para la generación automática de código de automoción de sistemas embebidos. El código se genera sobre la base de una máquina de estados finitos.

En [9] se presentó SolidFX, un marco de trabajo de ingeniería inversa para realizar análisis de código en C++, calcular métricas específicas en los sistemas, extraer datos de grandes bases de datos y analizar de manera visual e interactiva los resultados de la misma manera que se hace en la mayor parte de los IDE de desarrollo.

En [10] se presentó un marco de trabajo y una herramienta de edición como una extensión del lenguaje UIML. El desarrollador diseña su interfaz gráfica de usuario de

manera abstracta y posteriormente a través de la aplicación de técnicas de transformación basadas en gramáticas de grafos, dichas interfaces se transforman en interfaces de usuario concretas.

En [11] se propuso un nuevo enfoque el cual se basa en el desarrollo de una capa adicional Orientada a Aspectos que codifica un DSML (*Domain Specific Language*) para Frameworks basados en aplicaciones, de tal manera que se elimine la necesidad de implementar un generador de código.

En [12] se presentó un proceso para la generación de código fuente de RIAs (*Rich Internet Applications*) multi-dispositivo que complementa el proceso de Fases para el Desarrollo de RIAs denominado PPRD por sus siglas en inglés (*Phases Process for RIAs Development*).

En [13] se presentaron las herramientas Lonworks, KNX y BCU SDK Tools que ayudan a los desarrolladores a crear modelos de sistemas de automatización del hogar por medio de un lenguaje específico de dominio que se transforma en el código para plataformas de automatización del hogar específicas.

En [14] se describió una herramienta de generación de código para aplicaciones de lenguaje de procedimiento basado en el procesamiento distribuido. Los programas de aplicación, junto con las primitivas de partición se convierten en implementaciones concretas independientemente ejecutables.

En [15] se describió un método para automatizar el proceso de desarrollo de nuevas estructuras de datos, definiciones de meta-datos, y software de traducción a través de herramientas automatizadas. Con esto se generan automáticamente sistemas de colaboración.

La mayoría de las herramientas para el desarrollo o generación de aplicaciones encontradas en la literatura reportan diversas técnicas sin embargo no se observa que ofrezcan generar código a partir de técnicas de procesamiento de imágenes.

2.2. Técnicas de reconocimiento y procesamiento de imágenes

En [16] se presentó un estudio comparativo del procesamiento de imágenes de 3 diferentes técnicas: *Multiplicative Homomorphic Image Processing* (MHIP), *Log-Ratio Image Processing* (LRIP) y *Logarithmic Image Precessing* (LIP). En [17] se presentó un trabajo que contiene una revisión de los más recientes, así como de los clásicos métodos de registro de imágenes con el objetivo de proporcionar una fuente de referencia completa para los investigadores involucrados en el registro de imágenes, con independencia de las áreas de aplicación específicas. En [18] se presentó el sistema BioPro que emplea el método de Programación Orientada a imágenes también propuesto, el cual utiliza gráficos como una herramienta de diseño de software para aplicaciones Web.

En [19] se presentó el Framework AgentSketch para la interpretación de símbolos esbozados que explota en gran medida la información contextual para la resolución de ambigüedades en las imágenes. AgentSketch contempla bocetos de varios dominios uno de ellos los diagramas UML, específicamente de casos de uso.

En [20] se propuso un concepto y la arquitectura de un producto genérico basado en un reconocedor geométrico para la agrupación y segmentación de trazos. Reconoce

componentes individuales, diagramas esbozados como un conjunto y permite resolver las ambigüedades mediante análisis sintáctico y semántico.

En [21] se presentó un enfoque de reconocimiento visual y desarrollo de aplicaciones a partir de modelos para diagramas de ingeniería. El enfoque es una red neural convolucional aprovechada como un reconocedor de símbolos de ingeniería entrenable capaz de aprender las características visuales de las categorías de los símbolos definidos en algunos diagramas prototipo proporcionados por el usuario.

En [22] se discutió la noción de un diagrama de historia de estado. Una combinación de un metamodelado en UML y un marco de trabajo se utilizan para dar semántica precisa a los diagramas de la historia del estado y de los artefactos de forma orientada al análisis. Un diagrama de historia de estado o SHD es en realidad un diagrama de estados de transición.

En [23] se enfatizó el uso de *skeletons* en el procesamiento de imágenes digitales, para la realización de operaciones de procesamiento de imágenes, el esqueleto es una herramienta mucho más esencial y altamente adaptable. Los esqueletos son descriptores importantes en la representación de objetos y el reconocimiento.

En [24] se presentó un análisis de procesamiento de imagen paralela y distribuida con amplios detalles, se presentan resultados de un estudio de procesamiento de imágenes paralelo y distribuido con énfasis en los mecanismos, herramientas, tecnología, APIs utilizadas, dominios de aplicación y trabajos de investigación en curso.

Las diversas técnicas de reconocimiento y procesamiento de imágenes se reportan en diversos campos de aplicación, no obstante el momento no se han utilizado para cubrir las necesidades en la generación de código para aplicaciones multi-dispositivo. Por lo anterior, es necesario enfatizar la importancia del desarrollo de nuevos métodos que permitan crear aplicaciones de software de manera sencilla y rápida, y que posteriormente estos métodos se utilicen en herramientas para el desarrollo de aplicaciones.

3. Proceso de generación de software

En esta sección se describe el procedimiento de generación de software a partir del procesamiento de imágenes y reconocimiento de patrones.

Para el proceso de generación de software se utilizaron los siguientes algoritmos de procesamiento de imágenes:

Operaciones morfológicas: Un operador morfológico utiliza un elemento de estructuración para procesar una imagen. Las operaciones morfológicas se pueden utilizar en imágenes en binario y en escala de grises [25].

Operaciones de convolución: La convolución es una operación matemática que es fundamental para muchos operadores comunes de procesamiento de imágenes. Su campo de aplicación es amplio uno de ellos es el procesado lineal de imágenes [26].

Detección de bordes: Los bordes pueden ser detectados mediante la aplicación de un filtro de frecuencia de paso alto. Se utiliza ampliamente en la segmentación de la imagen cuando se quiere dividir la imagen en las zonas correspondientes a los diferentes objetos, llamadas zonas de interés [27].

Filtro de mediana: Un filtro de mediana es un filtro digital no lineal, que es capaz de preservar los cambios de señal agudos y es muy eficaz en la eliminación de ruido de impulso. Este algoritmo sustituye el valor de un pixel por el valor de la media de los pixeles vecinos. Es capaz de mejorar ciertas características de una imagen que posibiliten efectuar operaciones del procesado sobre ella [28].

El proceso de generación de software a partir del procesamiento de imágenes, consta de 8 pasos que a continuación se describen:

1) Entrada de la imagen a procesar, las imágenes deben estar delimitadas previamente, las imágenes soportadas son modelos de diseños ADVs (*Abstract Data Views*) que permiten especificar clara y formalmente interfaces de usuario separadas de los componentes de la aplicación de un sistema de software [29]. Los ADVs pueden ser generados por el usuario en cualquier herramienta de dibujo. Un ejemplo de estos ADVs se presentan posteriormente. Los formatos de imagen permitidos son: GIF, JPG o JPEG y PNG [30].

2) Validación de la imagen de entrada para asegurar que la imagen es una interfaz. La validación de la imagen consiste en aplicar los algoritmos de filtro de mediana, operaciones morfológicas, operaciones de convolución y detección de bordes para obtener los distintos elementos dentro de la imagen. Los elementos que se obtienen para las aplicaciones Web son: form, textarea, radiobutton, checkbox, list, por mencionar algunos. Los elementos que se obtienen para las aplicaciones para dispositivos móviles son: dataspinner, radiobuttongroup, spinnerlist, toggleswitch, button, checkbox, list, label, img, radiobutton, textarea, e input text. Si dentro de la imagen se localizan algunos de estos elementos entonces la imagen es válida para continuar con el proceso de generación de software.

3) Selección por parte del usuario de los elementos identificados que formarán parte del código a generar.

4) Selección del tipo de aplicación a generar: una aplicación Web o una aplicación móvil. Esta decisión se toma con la finalidad de que la generación de código se realice según las características de cada una de estas aplicaciones. Otro factor relevante que debe considerarse en esta fase son los elementos encontrados en la imagen de entrada, ya que no todas las imágenes tendrán elementos válidos para generar cualquier tipo de aplicación.

5) Configuración de la aplicación: de acuerdo con el tipo de aplicación seleccionada en el paso anterior, se selecciona la configuración general de la aplicación (título principal, lenguaje, color, descripción, entre otras).

6) Generación de un documento XML con etiquetas para: nombre de la aplicación, autor (s), etiquetas con la configuración de acuerdo a cada tipo de aplicación y lenguaje seleccionado esto es; resolución, orientación y elementos de plantilla (header, body, footer, entre otros); finalmente el conjunto de etiquetas con la representación de cada uno de los elementos seleccionados por el usuario.

7) Generación del código a partir de su representación XML: Se procesa el documento XML por medio de un documento XML Schema generando su equivalente en código HTML 5 de las etiquetas asociadas a cada elemento en la imagen, dicho código incrustado en frameworks tales como Sencha Touch un framework de HTML 5 que permite desarrollar aplicaciones web para diferentes dispositivos móviles [31]; JQuery Mobile un framework que permite agregar complejidad y enriquecer páginas [32]; y PhoneGap, un framework de desarrollo de aplicaciones web móviles que

permite a los desarrolladores construir aplicaciones web basadas en HTML5 y JavaScript con envoltorios para más de seis plataformas móviles, incluyendo iOS, Android y BlackBerry [33]. Por mencionar algunos; siguiendo con la descripción de este paso, la estructura generada se agrupa en clases de acuerdo con el lenguaje de programación seleccionado por el usuario.

8) Entrega al usuario de un archivo ZIP con el código fuente generado.

El proceso de generación de software se muestra a continuación en la Fig.1 donde se puede observar la secuencia de cada paso antes descrito.

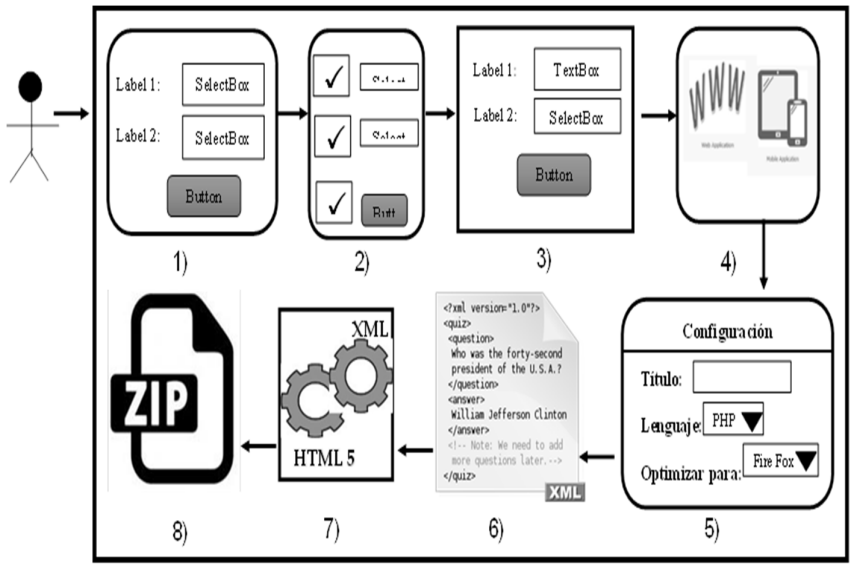

Fig. 1. Proceso de generación de software a partir del procesamiento de imágenes.

4. Proceso de identificación de elementos de interfaces

En principio se definen un conjunto de reglas para la identificación de cada elemento de la interfaz. La finalidad de definir el conjunto de reglas parte del hecho de tener un grupo de especificaciones para cada elemento que sirvan como identificadores únicos al momento de realizar el procesamiento de la imagen, dichas reglas están representadas en un árbol el cual emplea el lenguaje RuleML en su versión 1.0 [34].

En total se generaron 25 reglas considerando para ello 13 elementos como parte de una interfaz Web los cuales son: input text, button, checkbox, select, input date, input email, img, label, a (ancla), list, radiobutton, textarea y form. Para las interfaces para dispositivos móviles las reglas generadas son 12 y los elementos son: dataspinner, radiobuttongroup, spinnerlist, toggleswitch, button, checkbox, list, label, img, radiobutton, textarea, e input text. De acuerdo con la notación de RuleML 1.0 parte de los archivos XML generados para la representación de reglas, se muestran en la Fig. 2

donde, (a) es la representación de la regla para el elemento radiobutton de una interfaz Web, y (b) la regla para el elemento dataspinner de una interfaz móvil. RuleML es una iniciativa abierta en la que se busca establecer un sistema de reglas de inferencia lógica a partir de ontologías y documentos RDF con su propio lenguaje de especificación y ejecución.

```
<Implies>
        <If>
         <And>
          <Atom>
           <Var>Forma externa</Var>
           <Rel>Cuadrado</Rel>
          </Atom>
          <Atom>
           <Var>Figura interna</Var>
           <Rel>Círculo</Rel>
           <Ind>Posiblemente rellena</Ind>
          </Atom>
         </And>
        </If>
        <Then>
         <Atom>
          <Var>Forma externa</Var>
          <Var>Figura interna</Var>
          <Rel>Radiobutton</Rel>
         </Atom>
        </Then>
</Implies>
```

a)

```
<Implies>
     <If>
          <And>
           <Atom>
            <Var>Forma externa</Var>
            <Rel>Rectángulo</Rel>
            <Ind>Esquinas redondeadas</Ind>
           </Atom>
           <Atom>
            <Var>Figuras interna</Var>
            <Rel>Rectángulos</Rel>
            <Ind>6 en dos filas</Ind>
           </Atom>
          </And>
     </If>
     <Then>
          <Atom>
           <Var>Forma externa</Var>
           <Var>Figuras interna</Var>
           <Rel>Dataspinner</Rel>
          </Atom>
     </Then>
</Implies>
```

b)

Fig. 2. a) Regla para el elemento *radiobutton* de una interfaz Web, b) Regla para el elemento *dataspinner* de una interfaz móvil.

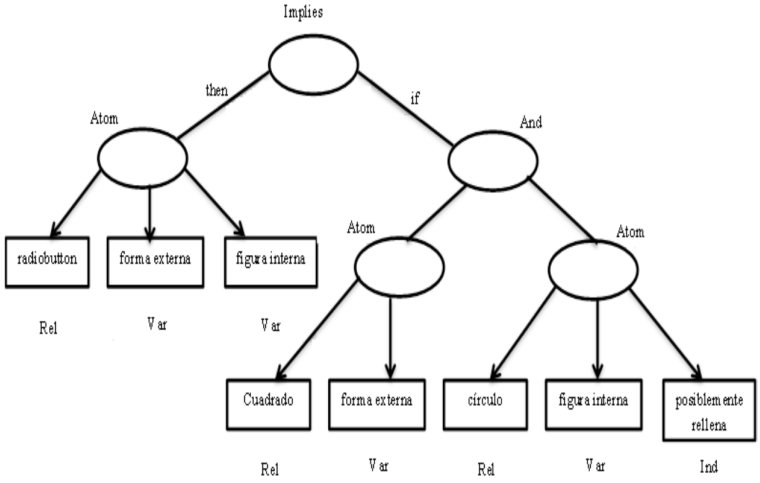

Fig. 3. Árbol parcial de reglas para el elemento *radiobutton* de una interfaz Web.

En la Fig. 3 se presenta un ejemplo del árbol de la regla definida para el elemento radiobutton de una interfaz Web. En la Fig. 4 se presenta el árbol de la regla para el elemento dataspinner de una interfaz móvil:

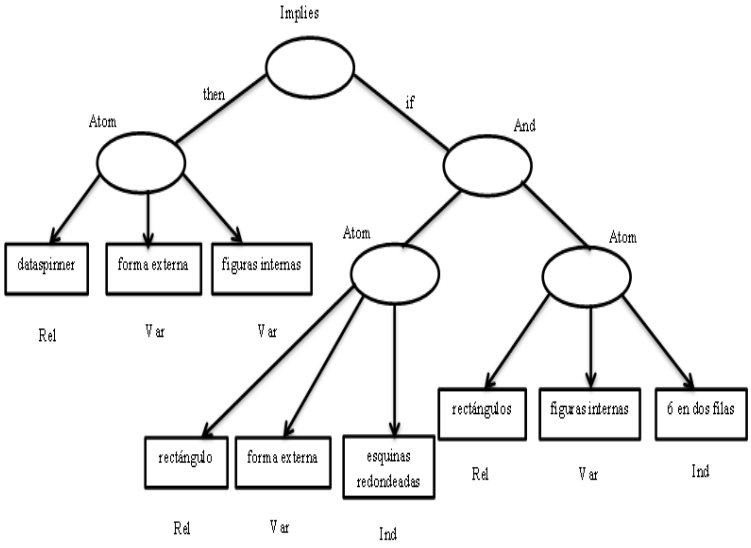

Fig. 4. Árbol parcial de reglas para el elemento *dataspinner* de una interfaz móvil.

Los arboles de reglas definidos en RuleML sirvieron como guía para su implementación en Matlab (*Matrix Laboratory*) [35], un software matemático que proporciona una biblioteca de funciones: *Image Processing Toolbox* para el procesamiento de imágenes. La *Image Processing Toolbox* de Matlab provee un conjunto de algoritmos bastante amplio, así como herramientas gráficas para procesamiento, análisis y visualización de imágenes digitales. Los principales algoritmos y funciones empleados para la identificación de elementos en el procesamiento de imágenes con Matlab son:

1) bwlabel: función que permite etiquetar los pixeles de cada objeto de manera única para su análisis, así como algunas de sus propiedades tales como la obtención de la imagen del objeto.

2) bwmorph: para realizar algunas operaciones morfológicas sobre la imagen del objeto.

3) edge: empleada para obtener los bordes de la imagen, a través del algoritmo de *Canny*.

4) houghlines: aplicada para la detección de líneas horizontales y verticales.

5) corner: permite obtener las coordenadas de las esquinas en una imagen.

5. Caso de estudio: generación de código a partir de una imagen que representa la portada principal de una página Web

En esta sección se presenta un caso de estudio como prueba de concepto del módulo propuesto. El caso de estudio representa la generación de código de la portada principal de una página Web a partir de la representación de su interfaz en una imagen. De acuerdo con las especificaciones establecidas para cada elemento se muestra a continuación un ejemplo que representan las especificaciones empleándose ya un diseño de interfaz. En la Fig. 5 se observa el ADV (*Abstract Data View*) de la portada principal de una página Web que consta de los siguientes elementos empezando de arriba hacia abajo: un elemento imagen que representa el banner, cuatro elementos ancla que representan un menú, un elemento label para dar la bienvenida, cuatro elementos imagen que representan un carrusel de imágenes y un elemento label con el pie de página.

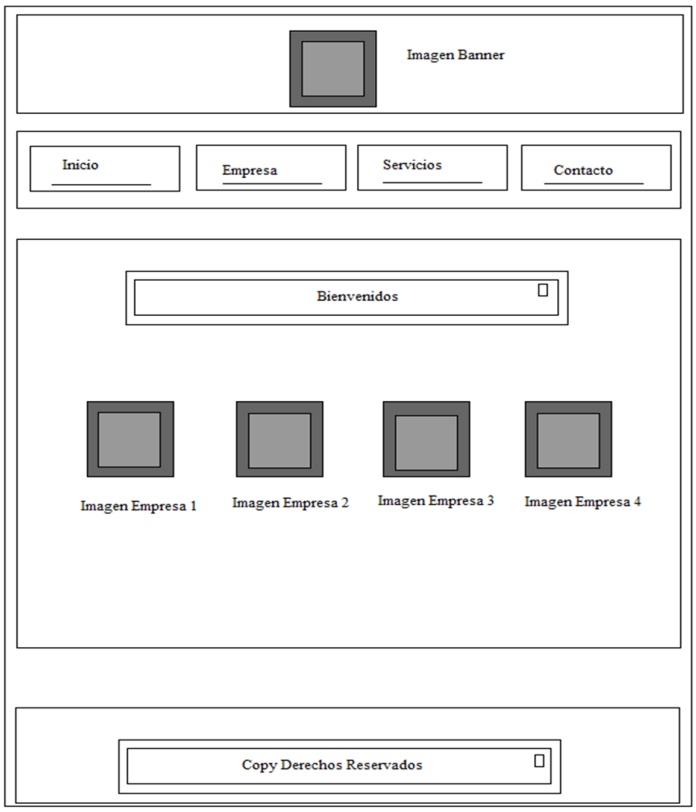

Fig. 5. Portada principal de una página Web.

A continuación se presenta la funcionalidad de un prototipo Web desarrollado en PHP que soporta el proceso de generación de software a partir del procesamiento de imágenes presentado en la Figura 1. Este prototipo es la herramienta generadora de código donde vista como una caja negra, el parámetro de entrada es una imagen y el parámetro de salida es una aplicación multi-dispositivo empaquetada. La herramienta emplea un módulo ejecutable desarrollado en Matlab para el procesamiento de la interfaz, este módulo se invoca desde una clase en Java, la cual esta encapsulada en un servicio web, y es invocado desde este cliente PHP para generar el código. Los requerimientos mínimos para ejecutar el módulo de procesamiento son un equipo con 6 GB en memoria RAM, un procesador Core (TM) i7 CPU 2.80 GHz, y un sistema operativo de 64 bits.

Detallando el funcionamiento de la herramienta tenemos en la Fig. 6 (a) el formulario para que el usuario seleccione la imagen que se someterá al procesamiento, en este caso la Fig. 5 es la imagen de entrada. Después de cargar la imagen se analiza con la finalidad de reconocer elementos dentro de la imagen, como resultado se muestra al usuario el conjunto de elementos detectados en ella como se ilustra en la Fig. 6 (b), de estos elementos el usuario selecciona todos aquellos que desea sean parte de su aplicación. El paso 3 de la Fig. 6 (c) indica al usuario algunas características de la imagen y los elementos que selecciono en el paso anterior y de los cuales se generará el código.

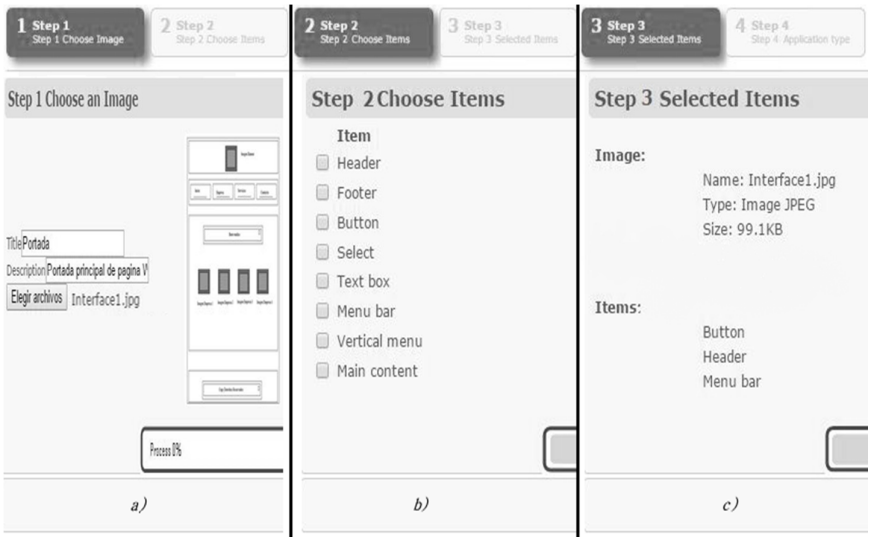

Fig. 6. a) Paso uno: selección de la imagen, b) Paso dos: selección de los elementos detectados, c) Paso tres: elementos seleccionados.

En la Fig. 7 (a) se muestra el paso 4 donde el usuario selecciona el tipo de aplicación que se generará: aplicación Web, aplicación de escritorio o aplicación móvil.

Fig. 7. a) Paso cuatro: selección del tipo de aplicación, b) Paso cinco: primera parte de la configuración de características generales de la aplicación.

Fig. 8. a) Paso cinco: segunda parte de la configuración de características generales de la aplicación y generación de código, b) Paso seis: descarga del código generado.

Después de seleccionar el tipo de aplicación en la Fig. 7 (b) el usuario configura algunos parámetros de acuerdo con el tipo de aplicación que seleccionó en el paso anterior. Para este ejemplo la configuración se muestra para una aplicación Web por lo que los parámetros solicitados son un título, el lenguaje de programación en que se debe generar el código, y el explorador para el que se desea optimizar la vista de nuestra interfaz.

En la Fig. 8 (a) el usuario configura algunas características más, tales como el color, ancho y alto, además de dar nombre y una pequeña descripción de la aplicación. Una vez enviados los datos de configuración se transforma cada elemento en su correspondiente código XML y se genera el código en el lenguaje seleccionado previamente. Finalmente en la Fig. 8 (b) el usuario puede descargar un archivo ZIP con el código en el lenguaje que seleccionó en los pasos anteriores.

6. Conclusiones y trabajo a futuro

Con base en la combinación de distintos algoritmos de procesamientos de imágenes se obtiene el reconocimiento de los distintos elementos que forman parte de una imagen representativa de una interfaz de usuario con lo que se tiene un primer aporte para una nueva y novedosa forma de desarrollar software.

Los aportes de este trabajo son prometedores en el área de la ingeniería de software en conjunción con técnicas de inteligencia artificial donde con la aplicación de un componente de software se apoyara a los desarrolladores facilitándoles la tarea de crear por ejemplo, prototipos de aplicaciones para presentarlas al usuario antes de iniciar con el desarrollo de una aplicación final, permitiendo así ahorrar tiempo y recursos económicos.

Como trabajo a futuro se pretende desarrollar una herramienta que permita a los usuarios generar las interfaces y por supuesto la generación automática del código de dicha interfaz. Además se espera aceptar que las imágenes provengan de bocetos hechos a mano o bien una cámara, por mencionar algunos ejemplos.

Por otra parte los resultados presentados aún son parciales con lo que se espera continuar con la puesta en marcha de distintos casos de estudio para evaluar correctamente el comportamiento del producto final, esto es; la correcta generación del código y porcentajes de error o eficiencia en el reconocimiento de elementos lo que permitirá continuar con la adaptación del procesamiento para otro tipo de imágenes.

Agradecimientos. Los autores agradecen el apoyado por el Consejo Nacional de Ciencia y Tecnología (CONACYT), Tecnológico Nacional de México (TecNM) y la Secretaria de Educación Pública (SEP) a través de PRODEP para la realización de este proyecto.

Referencias

1. De la Rosa, R.: Procesamiento de Imágenes Digitales. In: X Congreso Nacional en Informática y Computación del Instituto Tecnológico de Puebla (2007)
2. Eklund, A., Dufort, P., Forsberg, D., LaConte, S.M.: Medical image processing on the GPU–Past, present and future. Medical image analysis 17 (8), pp. 1073–1094 (2013)
3. Benoit, A., Caplier, A., Durette, B., Hérault, J.: Using human visual system modeling for bio-inspired low level image processing. Computer vision and Image understanding 114 (7), pp. 758–773 (2010)

4. Rué, F., Bijaoui, A.: A multiscale vision model applied to astronomical images. Vistas in Astronomy, 40(4), pp. 495–502 (1996)
5. Piszczek, M.: Laser Photography Examples of Processing of Image Information. Acta Physica Polonica, A, 124 (3), pp. 546–549 (2013)
6. Galván, J.V., Bertolino, S.R., Riveros, J.A.; Castellano, G.: Methodology for processing backscattered electron images. Application to Aguada archaeological paints. Micron 40 (8), pp. 793–799 (2009)
7. Obara, B.: An image processing algorithm for the reversed transformation of rotated microscope images. Computers & geosciences, 33(7), pp. 853–859 (2007)
8. Lindlar, F., Zimmermann, A.: A Code Generation Tool for Embedded Automotive Systems Based on Finite State Machines. In: Industrial Informatics, INDIN 2008, 6th IEEE International Conference, pp. 1539–1544 (2008)
9. Telea, A., Byelas, H., Voinea, L.: A Framework for Reverse Engineering Large C++ Code Bases. Electronic Notes in Theoretical Computer Science, Elsevier, pp. 143–159 (2009)
10. Iñesta, L., Aquino, N., Sánchez, J.: Framework and authoring tool for an extension of the UIML language. Advances in Engineering Software, Elsevier, pp. 1287–1296 (2009)
11. Santos, A., Koskimies, K., Lopes, A.: Automating the construction of domain-specific modeling languages for Object-Oriented Frameworks. The Journal of Systems and Software, Elsevier, pp. 1078–1093 (2010)
12. Colombo, M.L.O., Alor, H.G., Rodríguez, G.A.: A Novel Approach for Generating Multi-device Rich Internet Applications. In: CONIELECOMP, pp. 361–367 (2012)
13. Sánchez, P., Jiménez, M., Rosique, F., Álvarez, B., Iborra, A.: A Framework for developing home automation systems: From requirements to code. The Journal of Systems and Software, 84, Elsevier, pp. 1008–1021 (2011)
14. Sairaman, V., Ranganathan, N., Singh, N.: An Automatic Code Generation Tool for Partitioned software in Distributed Systems. In: Proceedings of the 19th International Conference on VLSI Design held jointly with 5th International Conference on Embedded Systems Design (VLSID '06), pp. 477–480 (2006)
15. Hartrum, T.: Automated Code Generation Tools for Collaboration Systems. Collaborative Technologies and Systems, International Symposium, pp. 183–190 (2007)
16. Pinoli, J.: A general comparative study of the multiplicative homomorphic, log-ratio and logarithmic image processing approaches. Signal Processing 58, Elsevier, 11–45 (1997)
17. Zitová, B., Flusser, J.: Image registration methods: a survey. Image and Vision Computing, Elsevier, pp. 977–1000 (2003)
18. Shimomura, T.: Visual design and programming for Web applications. Journal of Visual Languages and Computing, Elsevier, pp. 213–230 (2005)
19. Casella, G., Deufemia, V., Mascardi, V., Costagliola, G., Martelli, M.: An agent-based framework for sketched symbol interpretation. Journal of Visual Languages and Computing, 19, Elsevier, pp. 225–257 (2008)
20. Brieler, F., Minas, M.: A model-based recognition engine for sketched diagrams. Journal of Visual Languages and Computing, 21, Elsevier, pp. 81–97 (2010)
21. Luoting, F., Levent, K.: From engineering diagrams to engineering models: Visual recognition and applications. Computer-Aided Design, 43, Elsevier, pp. 278–292 (2011)
22. Draheim, D., Weber, G., Lutteroth, C.: Finite state history modeling and its precise UML-based semantics. In: Proceedings of the 2006 international conference on Advances in Conceptual Modeling: theory and practice (CoMoGIS'06), Springer-Verlag, pp. 43–52 (2006)
23. Komala, J., Punithavalli, M.: A survey on skeletons in digital image processing. In: International Conference on Digital Image Processing (ICDIP '09), IEEE Computer Society, Washington, DC, pp. 260–269 (2009)
24. Prajapati, H., Vij, S.: Analytical Study of Parallel and Distributed Image Processing. In: Image Information Processing (ICIIP) International Conference, pp. 1–6 (2011)

25. Vincent, L.: Morphological grayscale reconstruction in image analysis: applications and efficient algorithms. Image Processing, IEEE Transactions on, 2(2), pp. 176–201 (1993)
26. Socarrás, L. M. G., Sarmiento, A. J. C., Solano, S. S., Jiménez, P. B.: Diseño de bloques de convolución para procesado de imágenes con FPGA. Ingeniería Electrónica, Automática y Comunicaciones, 32(3), pp. 56–69 (2011)
27. Betancur, A., J. A., Prieto O., F. A., Osorio L., G. A.: Segmentación de frutos de café mediante métodos de crecimiento de regiones. Revista Facultad Nacional de Agronomía, Medellín, 59(1), pp. 3311–3333 (2006)
28. Gonzalez, R.C., Woods, R.E.: Digital Image Processing (3rd Edition). Prentice Hall (2007).
29. Cowan, D. D., Lucena, C. J. P. D.: Abstract data views: An interface specification concept to enhance design for reuse. Software Engineering, IEEE Transactions on, 21(3), pp. 229–243 (1995)
30. Ordoñez, S.C.A.: Formatos de imagen digital. Revista Digital Universitaria UNAM, 10 (10) (2005)
31. Clark, J. E., Johnson, B. P.: Sencha touch 2 mobile javascript framework. Packt Publishing Ltd (2013)
32. David, M.: Developing websites with jQuery mobile. Taylor & Francis (2011)
33. Ghatol, R., Patel, Y.: Beginning PhoneGap: Mobile Web Framework for JavaScript and HTML5. Apress (2012)
34. Boley, H., Paschke, A., Shafiq, O.: RuleML 1.0: the overarching specification of web rules. Lecture Notes in Computer Science, 6403(4), pp. 162–178 (2010)
35. Guide, M. U. S.: The mathworks. Natick, MA, 5, 333 (1998)