GigMaleBPMN: Generation of Graphical Components from the BPMN Model using Machine Learning

Marco_Antonio de_la _Cruz
Universidad Peruana de Ciencias Aplicadas, Prolongación Primavera 2390, Lima 15023 - Perú
u201422580@upc.edu.pe

Luis_Miguel Estrada
Universidad Peruana de Ciencias Aplicadas, Prolongación Primavera 2390, Lima 15023 - Perú
u20181d335@upc.edu.pe

Eduardo Díaz
Universidad Peruana de Ciencias Aplicadas, Prolongación Primavera 2390, Lima 15023 - Perú
pcsijord@upc.edu.pe

José Ignacio Panach
Escola Tècnica Superior d'Enginyeria, Departament d'Informàtica Universitat de València,
Avenida de la Universidad, s/n, 46100, Burjassot, València, Spain.
joigpana@uv.es

***Abstract*— The BPMN model allows organizations to depict business processes. However, this model does not capture the functional behavior of the system to generate graphical components. This article presents GigMaleBPMN a method for generating graphical components from a BPMN model using Machine Learning. The method is structured in four steps: (1) creating a BPMN model, (2) using Machine Learning to identify the elements of the BPMN model and indicate which graphical components should be used, (3) manually developing wireframes in Balsamiq based on the identified BPMN elements, and (4) using Machine Learning to identify the graphical components of the wireframes, enabling the automatic generation of graphical components and code. To enhance understanding, an illustrative example was developed using the method. The results prove that this approach allows for the automatic generation of graphical components using Machine Learning from a BPMN model.**

Keywords—BPMN, Machine Learning, Wireframes, Graphics components.

I. INTRODUCTION

A BPMN model (Business Process Model and Notation) is used to describe and model an organization's internal processes, allowing for understanding of its internal procedures in a graphical format and standardized communication. [1]. A BPMN model includes basic elements: *events*, *tasks*, *gateways* and *flows*, *artifacts*, *sub processes*, and others [2]. Additionally, there are two types of *tasks* that are commonly used: user tasks (involving user interaction with the software) and service tasks (linking to a web service or an automated application) [3]. However, BPMN models do not allow capturing the functional behavior of the system to generate graphical components [4]. BPMN models are developed by business analysts, who are not the ones involved in the design of the graphical components. As a result, the final implementation is not entirely correct [5]. Therefore, developing a BPMN model alone is not sufficiently useful for creating graphical components, as the developer would spend more time understanding the business process rather than designing the graphical components [6].

On the other hand, there are tools that allow generating graphical components from a BPMN model, such as Bizagi [7], Aura Portal [8], Bonitasoft [9], E-citiz Studio [10], where other conceptual models are used to complement the BPMN model. However, the widespread problem in the presented works falls on the strong dependency that the methods have when extending BPMN models for generating graphical user interfaces.

Likewise, the use of Machine Learning is increasingly prevalent in society [11], [12], [13], due to the various disciplines where its techniques can be applied [14] and the accuracy with which it performs a specific activity. In fact, techniques for object detection within an image are commonly applied in tasks where a minimum error percentage is needed [15]. Therefore, Microsoft Azure supplies diverse services to use different Machine Learning techniques, such as Computer Vision [16] or Custom Vision [17], specialized services in image recognition techniques [18]. Furthermore, there are works that use Machine Learning to generate graphical components, such as [19], [20], [21], allowing them to recognize images and generate graphical components.

The contribution of this work is the development of a method called GigMaleBPMN, which stands for "Graphical Interface Generator using Machine Learning and BPMN" This method enables the automatic generation of graphical components from a BPMN image using Machine Learning techniques. The method is structured into four steps: (i) creating the BPMN model in the Bizagi tool, (ii) identifying the BPMN elements using Machine Learning techniques to determine which graphical components should be developed, (iii) creating wireframes [22] based on the BPMN elements using the Balsamiq tool [23], and (iv) identifying the graphical components from the wireframes using Machine Learning techniques to automatically generate graphical components. These graphical components can be generated in JavaScript, Angular, React, and Vue [24] using the Google Material Design system [25]. The main objective is the ability to save time and money for organizations by quickly generating graphical components automatically [26]. Additionally, designers often create low-fidelity wireframes before

designing graphical components, as it allows them to have a software artifact resembling graphical components in a short amount of time [27].

The rest of the paper is structured as follows. Section 2 presents related works to our proposal. Section 3 presents the method proposed by the researchers for generating graphical components from a BPMN model using Machine Learning. Section 4 supplies an illustrative example. Finally, Section 5 presents some conclusions and future work.

II. State of the Art

In this section, we review works related to the generation of graphical components from a BPMN model using Machine Learning techniques. To achieve this, we conducted a Targeted Literature Review (TLR), a non-systematic, comprehensive, and informative bibliographic review, to gather relevant references. The semantic question related to this topic is translated into the following syntactic query used as a search string in the Scopus digital library [28]:

"BPMN" AND ("Machine Learning" OR "user interface" OR "graphical components"). The articles will be classified into two groups: (i) Generation of graphical components from a BPMN model, and (ii) Use of Machine Learning techniques in the generation of graphical components. The inclusion criteria are: (1) Generation of graphical components from a BPMN model, and (2) Use of Machine Learning techniques to generate graphical components. The exclusion criteria are: (1) Unrelated topics to the BPMN model, (2) Approaches that do not generate graphical components from a BPMN model, and (3) Approaches that do not generate graphical components using Machine Learning techniques. The first search yields 110 scientific articles. After applying the inclusion and exclusion criteria, a sample of 8 articles has been considered. The following are the accepted and grouped articles from the search:

A. Generation of graphical components from a BPMN model

Díaz et al. [29] [6] present a method to generate graphical components from BPMN models. The first research is based on the analysis of eighteen real BPMN projects and five BPMN patterns to decide transformation rules from BPMN models to graphical components. The method uses a list of stereotypes for the transformation rules with more than one design alternative. The main problem is the strong dependency between BPMN models and class diagrams. One limitation is the focus on a limited set of stereotypes, and further studies would be necessary to evaluate the effectiveness of the tool with a larger set of stereotypes and more complex BPMN models.

Brambilla et al. [30] focuses on the use of the WebRatio BPM (Business Process Model) tool for designing web applications based on business processes. It is based on the model-driven development approach and uses model transformations to automatically generate web applications from business process models. The method involves the extension of three main components of the WebRatio tool suite: the model editor, the code generator, and the runtime libraries.

Lei Han et al. [31] proposes a framework for the automatic derivation of user interfaces from BPMN models. The method used is based on creating business process models with added roles and data relationships, which are then used to create user interfaces. The main problem faced by the proposed method is its heavy reliance on the quality of the first system requirements specification. Additionally, this approach focuses on the design phase and does not supply guidance on how to implement the resulting system. Therefore, it may be necessary to complement it with other approaches to achieve an effective system implementation.

Gonzalez et al. [32] derives software analysis and design from BPMN models. The method is based on applying correction, completion, automation, and optimization patterns to BPMN models to create analysis and design models in the Unified Modeling Language (UML) [33], using the use of a rule engine like Drools [34] to implement model transformations. Its main problem is the complexity and heterogeneity of BPMN business process models, which makes their correction and optimization challenging.

To summarize related Works considered in this sub-section, we can state that they generated graphics components from BPMN models [29], [6], [30], [31], [32]. The widespread problem in all of them is that the quality of the generated interfaces depends on the quality of the extended or transformed BPMN model, as well as the first specification of system requirements. Additionally, most of the works have limitations in terms of complexity. Therefore, our proposal is based on generating graphical components without extending the BPMN model, using Machine Learning techniques that have been trained under certain conditions.

B. Use of Machine Learning techniques in the generation of graphical components

Moran et al. [19] focuses on automating the transformation of graphical components into source code for mobile applications. For this purpose, it relies on three tasks: detection, classification, and assembly. The method consists of integrating these steps: (i) computer vision [35] to detect the logical components of the GUI from the sketch, (ii) automated dynamic analysis along with software repository mining techniques to accurately classify different components according to their domain, such as toggle buttons, among others, (iii) a data-driven nearest neighbor-based algorithm [36] to achieve the assembly of the graphical components' structure.

Nguyen et al. [20] addresses the problem of automatically generating the user interface (UI) code for mobile applications from screenshots or conceptual drawings. For this purpose, the author created the technique called Reverse Engineer Mobile Application User Interfaces (REMAUI), which uses Machine Learning techniques such as computer vision and optical character recognition (OCR) [37], to identify elements of the user interface, such as images, text, containers, and lists, from an input image.

Chen et al. [21] uses a neural translator to convert a design image into a skeleton of graphical components. They combined advancements in computer vision and automatic translation into a unified neural network framework [38]. The proposed method is based on learning visual features in graphical part images, encoding the spatial arrangement of these features, and the automated generation of skeletons. Additionally, an automated graphical component exploration method is developed to gather large-scale user interface data from real applications, which is used to train the neural translator.

As a conclusion on Works considered in this sub-section, we can state that there is proposal on Machine Learning techniques [19], [20], [21] for the automatic generation of graphical components. However, the main issue with all of them is that if the detection or classification of the graphical components from the user's sketch is not correct, the result may be affected in terms of fidelity or functionality.

III. Method to Transform from BPMN Model to Graphical Components

This section presents the development of the GigMaleBPMN method, which stands for "*Graphical Interface Generator using Machine Learning and BPMN*", it allows the generation of graphical components from a BPMN model using Machine Learning with the use of advanced image and text detection algorithms. This method is divided into four steps, with each step explained in the following subsections:

(1) Elaborate BPMN model: The analyst creates the BPMN model using the Bizagi tool.

(2) Identify BPMN elements using Machine Learning: The resulting BPMN model serves as input for using Microsoft Azure's Machine Learning to detect different elements of the BPMN model and decide which graphical component needs to be developed.

(3) Wireframes development: The analyst designs the wireframes (visual representation of the graphical components) that will incorporate the BPMN elements. The Balsamiq tool [23] is used for creating the wireframes.

(4) Generate graphical components: The result of the wireframes serves as input for using Microsoft Azure's Machine Learning to detect the elements of the developed wireframes. This enables the generation of graphical components in JavaScript technologies such as Angular, React, or Vue.

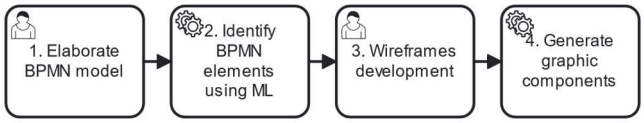

Fig. 1. Steps of the GigMaleBPMN method.

Figure 1 shows the four steps of the GigMaleBPMN method. Step 1 involves developing the BPMN model in the Bizagi tool. In Step 2, the GIG (Graphical Interface Generator) tool, developed by the researchers using Microsoft Azure services, is used to name the BPMN elements and show which graphical components should be developed. In Step 3, the analyst creates wireframes for each BPMN element using the Balsamiq tool. In Step 4, the GIG tool allows for the automatic identification of elements in the wireframes to generate graphical components such as forms, textboxes, check buttons, list boxes, combo boxes, and others.

A. Step 1: Elaborate BPMN model

In this first step, the BPMN model is developed by the analyst. This model has the business processes, which are represented with BPMN elements such as lanes, events, tasks (user tasks, service tasks, and others), gateways (exclusive, parallel, and others), subprocesses, and other elements [7]. The modeling tool that will be used for creating the BPMN model is Bizagi (v. 11.2.5) [7] which is one of the most widely used by organizations [39]. Figure 2 shows an example of a BPMN model depicting the purchase order generation process. The "Purchase Department" lane has the service task called "Create Purchase Order" and the user tasks "Send Order to ERP" and "Update ERP", as well as parallel gateways and an exclusive gateway.

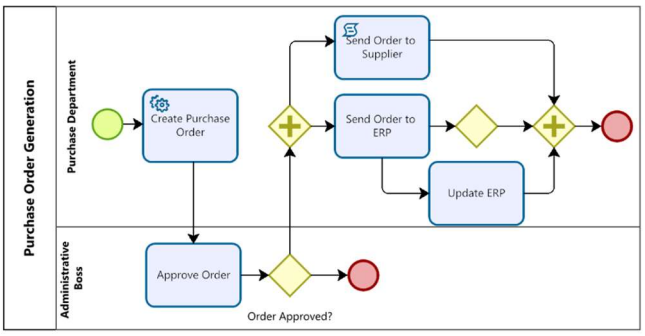

Fig. 2. Example of a BPMN Model on the Purchase Order Generation Process.

B. Step 2: Identify BPMN elements using Machine Learning

For each BPMN model built in step 1, the BPMN elements must be identified using Machine Learning provided by Microsoft Azure's Custom Vision and Computer Vision tools. For this step, the Graphical Interface Generator (GIG) tool was developed, and its development process is as follows:

(i) User registration is required in the Microsoft Azure Custom Vision portal, which allows managing multiple projects and using services such as Machine Learning training:

(ii) In the Microsoft Azure service section, forty images of BPMN models were added (these BPMN models are from real-world projects in educational, business, technology, healthcare, and other contexts), highlighting their BPMN elements. One BPMN model was selected to assign labels to the BPMN elements. These labels will be used for the Microsoft Azure tool to recognize each BPMN element. The following BPMN elements were labeled: (a) user task was assigned the label "user-task," (b) service task was assigned the label "service-task," (c) exclusive gateway was assigned the label "exclusive-gateway," (d) parallel gateway was assigned the label "parallel-gateway." This will allow naming each BPMN element. Figure 3 shows the labeling of the elements in a BPMN model:

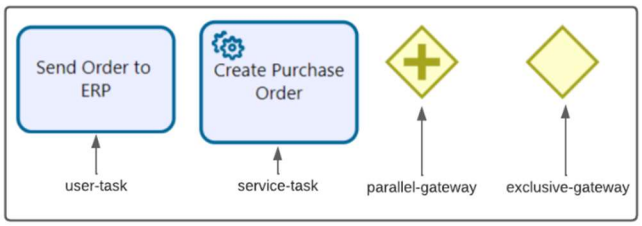

Fig. 3. Labeling of the BPMN elements image.

(iii) The Microsoft Azure Custom Vision tool has a "Train" option, which is used to train all the images of the loaded BPMN models, where a labeled BPMN model is trained. Training involves executing a Machine Learning algorithm provided by Microsoft Azure to predict which BPMN elements have the labels in the image. Custom Vision uses machine learning algorithms, including Convolutional Neural Network (CNN) [40], to train a model that can

recognize the registered labels. During training, the training time and neural network connections are adjusted to optimize its generalization ability and correct classification [17].

(iv) After training the tool, the REST API service [41] needs to be integrated. In our case, using the Java language, we use HTTP Request [41] for prediction to recognize the BPMN elements. The response, in this case, is returned in JSON code [42], providing data such as: (i) label name, (ii) probability (a numerical value showing the accuracy percentage with the BPMN elements trained by the Azure platform), (iii) positions of the BPMN elements found in the image. Figure 4 shows the schema of the JSON query result for predicting an image with a BPMN model. It includes "tagname" as the label name, "probability" as the probability percentage compared to what the tool has been trained on, and "boundingbox" as the position and size of the detected section in the overall image.

application/json application/xml text/xml

Sample Schema


```
{
  "id": "00000000-0000-0000-0000-000000000000",
  "project": "00000000-0000-0000-0000-000000000000",
  "iteration": "00000000-0000-0000-0000-000000000000",
  "created": "string",
  "predictions": [
    {
      "probability": 0.0,
      "tagId": "00000000-0000-0000-0000-000000000000",
      "tagName": "string",
      "boundingBox": {
        "left": 0.0,
        "top": 0.0,
        "width": 0.0,
        "height": 0.0
      },
      "tagType": "Regular"
```


Fig. 4. JSON result of the Custom Vision API query for detecting components in a BPMN model image.

On the other hand, in Computer Vision, Optical Character Recognition (OCR) [16] techniques based on Machine Learning allow extracting printed or handwritten text from images [16]. To use this service on the Microsoft Azure platform, an instance needs to be created in the Computer Vision service, which supplies the necessary credentials for making queries to the REST API. Here's an example of the syntax for the link: "https://{instance-name}.cognitiveservices.azure.com/vision/v3.2/ocr?language=es&detectOrientation=true&model-version=latest", where the image is sent (in this case, images with previously detected task texts) along with the subscription key provided when creating the instance. The returned model is shown in Figure 5, which displays the detected text in each image classified by its label type.


```
{ "language": "en",
  "textAngle": -2.0000000000000338,
  "orientation": "Up",
  "regions": [
    {
      "boundingBox": "462,379,497,258",
      "lines": [
        {
          "boundingBox": "462,379,497,74",
          "words": [
            {
              "boundingBox": "462,379,41,73",
              "text": "A"
            }
```


Fig. 5. JSON result of the word detection in the BPMN model image using Computer Vision API.

When the object detection is complete, it returns a JSON response to the GIG tool's requesting service, which has several objects with different properties such as tagName, boundingBox (left, top, width, and height), and probability. With this information, the BPMN element detection flow can continue.

Figure 6 shows the flow, where the GIG tool makes an HTTPS request to the Microsoft Azure services, and Microsoft Azure returns a JSON code of the detected elements.

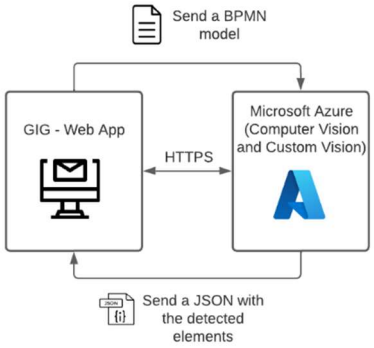

Fig. 6. Flow showing the interaction between the GIG tool and Microsoft Azure.

The identified BPMN elements will be the descriptions of user tasks, service tasks, exclusive gateways, and parallel gateways, which will supply the resulting graphical components that need to be developed.

C. Step 3: Wireframes development

After the GIG tool names the BPMN elements such as user tasks, service tasks, exclusive gateways, and parallel gateways from the BPMN model, this step 3 continues with wireframe development. The wireframes are created using the Balsamiq tool, which allows the integration of graphical components in a simple and user-friendly manner [23]. For wireframe creation, it is important to use only the following graphical components: Textbox, Block Text, Combo box, Title, Checkbox, Button, Data Table, Label, as other components are not included in the scope of this work.

Subsequently, for each detected BPMN element from the BPMN model, the user should create wireframes of the suggested graphical components from step 1, and these wireframes should be saved in JPG or PNG format. Figure 7 shows an example wireframe developed in the Balsamiq tool, featuring graphical components such as labels, input search, and buttons:

Fig. 7. Example of a wireframe developed in the Balsamiq tool.

D. Step 4: Generate graphical components

With the wireframes developed in step 3, the next step is to upload the wireframes in PNG or JPG format so that the GIG tool can recognize the graphical components of the wireframe. The GIG tool needs to be configured, and this process was conducted as follows:

(i) In the Microsoft Azure service area of the GIG tool, forty wireframes in PNG or JPG format were added with their graphical components.

(ii) An image of a wireframe holding graphical components was selected, and labels were assigned to the following graphical components: (a) "button" for push button, (b) "label" for titles, (c) "input" for textbox, (d) "table" for data table, (e) "checkbox" for checkbox, and (f) "select" for combo box. Figure 8 shows the assignment of labels to the graphical components, to then continue with the same process from step 2, which is to name elements such as graphical components.

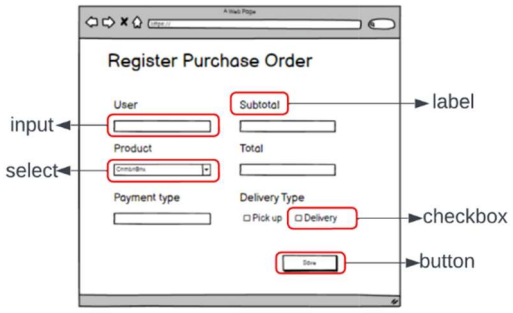

Fig. 8. Label assignment to the graphical components of the wireframe.

(iii) When the graphical components of a wireframe are detected, it will return a JSON response with information about that component (element type, dimensions, and label name). For example, if it recognizes a push button, it will display the following information: label: "button" and probability: 95%, which shows the estimated probability that the object or image belongs to the specified label. Then, the generation of graphical components can continue.

Finally, with this information, graphical components are generated based on the identified tasks and gateways from step 2 and the wireframes designed in step 3. The GIG tool allows downloading the source code in HTML, CSS, and JavaScript technologies such as React, Vue, and Angular [24].

IV. Illustrative Example of the Graphical Component Generator from a BPMN Model

This section presents an illustrative example of the GigMaleBPMN method. For this example, the BPMN project from the Bizagi repository called "Purchase Order Creation" was used. The project proves a flow for registering a purchase order started by the requested and received by the administrative manager. This process flow includes various tasks and gateways used to confirm all necessary steps.

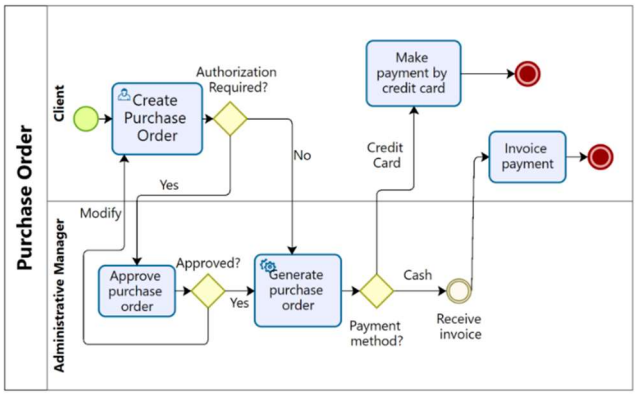

Fig. 9. BPMN model of the "Purchase Order" project.

Figure 9 shows that the "Create Purchase Order" user task starts in the customer lane, responsible for registering the purchase order data, and continues with the "Require Authorization" gateway that confirms the order creation. It has two paths: if it is "Yes," the "Approve Purchase Order" user task continues, and if it is "No," the "Generate Purchase Order" service task continues. The latter refers to a record by the administrator with added data to the earlier registration made by the purchasing department. It then goes through the "Payment Method" gateway. If it is paid by credit card, the "Make Credit Card Payment" task is conducted, and the process ends. If there is no credit card, the order invoice is received to try with "Invoice Payment" using cash, concluding the purchase order process. This BPMN model should be imported as an image in PNG or JPG format in the next step, so that it can recognize which graphical components need to be developed.

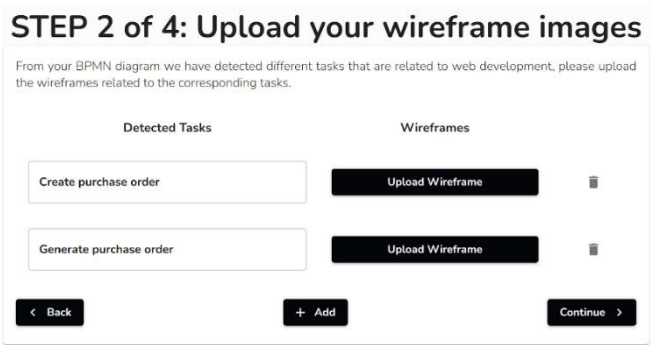

Fig. 10. Result of task detection from the BPMN model.

Figure 10 shows the result of applying the tool to process the imported image of the BPMN from the "Purchase Order Creation" project. The GIG tool detected the "Create Purchase Order" user task and the "Generate Purchase Order" service task. Therefore, wireframes need to be developed, supported by the UML class diagram used by the analyst to input the attributes, relating them to each detected task. To import each wireframe, click on the "Upload Wireframe" option, as you will not be able to continue if each task does not have its own wireframe.

In this example, we used Figure 7, which is the "Register Purchase Order" task and includes attributes such as user, subtotal, product, total, and payment type. To develop this wireframe, the analyst relied on the class diagram to determine which attributes were necessary. The next step is to try generating a graphical user interface.

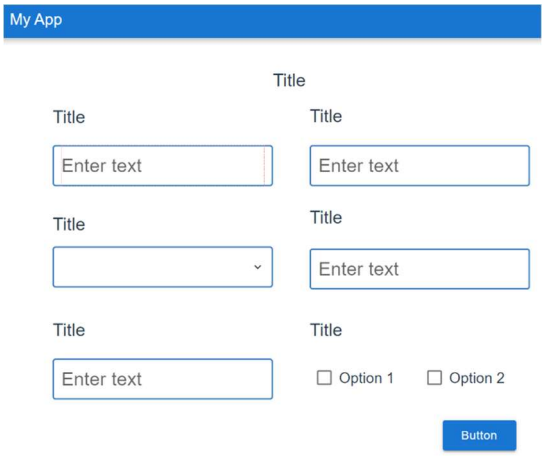

Fig. 11. Graphical components of the "Register Purchase Order" user task.

Figure 11 shows the result of the graphical components. The steps of the GigMaleBPMN method were applied to the "Purchase Order" BPMN model. Different fields were created for the identified labels, like Textbox for input, Title for label, Checkbox for checkbox, Button for button, and Combo box for select.

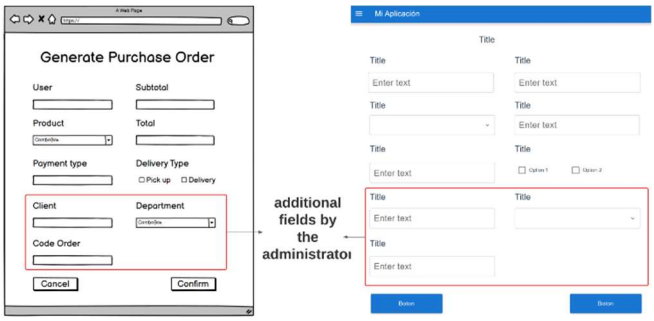

Fig. 12. Graphical components of the "Generate Purchase Order" user task that the administrator sees.

Figure 12 shows another example of the result of the graphic components detected in the “Generate Purchase Order” service task, creating six text boxes, two combo boxes, two check boxes, ten titles and two buttons on the page of this task. Likewise, the application of the GigMaleBPMN method can generate projects in frameworks like Angular and Vue, as well as the React library.

V. Conclusions

This article presents the GigMaleBPMN method, which enables the generation of graphical components from a BPMN model using Microsoft Azure's Machine Learning techniques.

The GigMaleBPMN method consists of four steps: (i) creating the BPMN model in Bizagi, (ii) naming BPMN elements to show which graphical components should be developed, (iii) developing wireframes based on the mentioned graphical components in step 2, and (iv) naming the developed graphical components in the wireframes, allowing for their generation. The BPMN elements used in the GigMaleBPMN method are user tasks, service tasks, exclusive gateways, and parallel gateways. The graphical components that the GigMaleBPMN method can generate include Input, Combo box, Text, Checkbox, Button, Data Table, and Label from the wireframe. The result is graphical components in source code form for various programming language technologies such as JavaScript (Angular, Vue, and React).

The work supplies an illustrative example of the process using a BPMN project from the Bizagi repository called "Purchase Order Creation." This example serves to better understand the process from BPMN element detection to graphical component generation.

The limitations of this work are as follows: (i) it only recognizes user tasks, service tasks, parallel gateways, and exclusive gateways; (ii) only 40 BPMN models were used; (iii) there is a dependency on using the class diagram when developing wireframes; (iv) it uses Microsoft Azure's Machine Learning algorithms.

Future work includes: (i) naming more BPMN elements, (ii) generating more graphical components, and (iii) naming BPMN patterns found in BPMN models.

References

[1] “BPMN Specification - Business Process Model and Notation.” https://www.bpmn.org/ (accessed Jun. 09, 2023).

[2] “View the BPMN Quick Guide - BPMN Quick Guide.” https://www.bpmnquickguide.com/view-bpmn-quick-guide/ (accessed Jun. 09, 2023).

[3] “BPMN Modeling and Reference Guide Digital Edition | Enhanced Reader.”

[4] E. F. Cruz and A. M. Rosado Da Cruz, “Deriving integrated software design models from BPMN business process models,” in *ICSOFT 2018 - Proceedings of the 13th International Conference on Software Technologies*, SciTePress, 2019, pp. 571–582. doi: 10.5220/0006852006050616.

[5] E. Díaz, J. I. Panach, S. Rueda, and J. Vanderdonckt, “An empirical study of rules for mapping BPMN models to graphical user interfaces,” *Multimed Tools Appl*, vol. 80, no. 7, pp. 9813–9848, Mar. 2021, doi: 10.1007/s11042-020-09651-6.

[6] E. Díaz, J. I. Panach, S. Rueda, and D. Distante, “A family of experiments to generate graphical user interfaces from BPMN models with stereotypes,” *Journal of Systems and Software*, vol. 173, Mar. 2021, doi: 10.1016/j.jss.2020.110883.

[7] “Bizagi, One Platform; Every Process. User Guide Studio.” https://help.bizagi.com/bpm-suite/en/index.html?bpmn_shapes.htm (accessed May 30, 2023).

[8] “Aurora Portal.” http://tiacws.com/aurora/ (accessed Jun. 13, 2023).

[9] “Bonitasoft : Open Source BPM software - Business Process Management.” https://www.bonitasoft.com/ (accessed Jun. 13, 2023).

[10] “Gestion de e-procédure grâce au BPM e-Citize-Citiz_.” https://www.e-citiz.com/bpm (accessed Jun. 14, 2023).

[11] “Top 10 Applications of Machine Learning | Daily Life Applications | Edureka.” https://www.edureka.co/blog/machine-learning-applications/ (accessed Jun. 20, 2023).

[12] T. H. Hu, L. Wan, T. A. Liu, M. W. Wang, T. Chen, and Y. H. Wang, “[Advantages and Application Prospects of Deep Learning in Image Recognition and Bone Age Assessment].,” *Fa Yi Xue Za Zhi*, vol. 33, no. 6, pp. 629–634, Dec. 2017, doi: 10.3969/J.ISSN.1004-5619.2017.06.013.

[13] F. Recknagel, “Applications of machine learning to ecological modelling,” *Ecol Modell*, vol. 146, no. 1–3, pp. 303–310, Dec. 2001, doi: 10.1016/S0304-3800(01)00316-7.

[14] “What is Machine Learning? | How it Works, Tutorials, and Examples - MATLAB & Simulink.” https://www.mathworks.com/discovery/machine-learning.html (accessed Jun. 20, 2023).

[15] P. Bhushan *et al.*, “A Self-Attention Based Hybrid CNN-LSTM Architecture for Respiratory Sound Classification,” *GMSARN International Journal*, vol. 18, no. 1, pp. 54–61, 2024.

[16] “OCR - Optical Character Recognition - Azure Cognitive Services | Microsoft Learn.” https://learn.microsoft.com/en-us/azure/cognitive-services/computer-vision/overview-ocr (accessed May 30, 2023).

[17] “What is Custom Vision? - Azure Cognitive Services | Microsoft Learn.” https://learn.microsoft.com/en-us/azure/cognitive-services/custom-vision-service/overview (accessed May 30, 2023).

[18] “What are Azure Cognitive Services? - Azure Cognitive Services | Microsoft Learn.” https://learn.microsoft.com/en-us/azure/cognitive-services/what-are-cognitive-services (accessed Jun. 20, 2023).

[19] K. Moran, C. Bernal-Cardenas, M. Curcio, R. Bonett, and D. Poshyvanyk, “Machine Learning-Based Prototyping of Graphical User Interfaces for Mobile Apps,” *IEEE Transactions on Software Engineering*, vol. 46, no. 2, pp. 196–221, Feb. 2020, doi: 10.1109/TSE.2018.2844788.

[20] T. A. Nguyen and C. Csallner, “Reverse engineering mobile application user interfaces with REMAUI,” in *Proceedings - 2015 30th IEEE/ACM International Conference on Automated Software Engineering, ASE 2015*, Institute of Electrical and Electronics Engineers Inc., Jan. 2016, pp. 248–259. doi: 10.1109/ASE.2015.32.

[21] C. Chen, T. Su, G. Meng, Z. Xing, and Y. Liu, “From UI design image to GUI skeleton: A neural machine translator to bootstrap mobile GUI implementation,” *Proceedings - International Conference on Software Engineering*, pp. 665–676, May 2018, doi: 10.1145/3180155.3180240.

[22] “What Is A Wireframe? Your Best Guide.” https://careerfoundry.com/en/blog/ux-design/what-is-a-wireframe-guide/#what-is-a-wireframe (accessed Jun. 06, 2023).

[23] “The best wireframe tools in 2023 | Creative Bloq.” https://www.creativebloq.com/wireframes/top-wireframing-tools-11121302 (accessed May 30, 2023).

[24] “Angular vs React vs Vue: Core Differences | BrowserStack.” https://www.browserstack.com/guide/angular-vs-react-vs-vue (accessed May 30, 2023).

[25] “Material Design.” https://m3.material.io/ (accessed Jun. 10, 2023).

[26] “What is a Graphical User Interface (GUI)? - Definition from Techopedia.” https://www.techopedia.com/definition/5435/graphical-user-interface-gui (accessed Jun. 06, 2023).

[27] “Why Wireframes Are Important in the Design Process. | by Proto.io | Medium.” https://protoio.medium.com/why-wireframes-are-important-in-the-design-process-de4e773e611 (accessed Jun. 06, 2023).

[28] “Scopus preview - Scopus - Welcome to Scopus.” https://www.scopus.com/home.uri (accessed May 30, 2023).

[29] E. Diaz, J. I. Panach, S. Rueda, and O. Pastor, “Towards a Method to Generate GUI Prototypes from BPMN.”

[30] M. Brambilla, S. Butti, and P. Fraternali, “WebRatio BPM: A Tool for Designing and Deploying Business Processes on the Web.”

[31] J. Cao, X. Liu, and K. Ren, Eds., *Process-Aware Systems*, vol. 602. in Communications in Computer and Information Science, vol. 602. Singapore: Springer Singapore, 2016. doi: 10.1007/978-981-10-1019-4.

[32] E. Aïmeur, U. Ruhi, and M. Weiss, Eds., *E-Technologies: Embracing the Internet of Things*, vol. 289. in Lecture Notes in Business Information Processing, vol. 289. Cham: Springer International Publishing, 2017. doi: 10.1007/978-3-319-59041-7.

[33] “OMG | Object Management Group.” http://www.omg.org/ (accessed Jun. 13, 2023).

[34] B. Dong, Y. Sun, W. Chen, X. Xu, and Y. Zhang, “Dynamic environment monitoring system of digital twin computer room based on Drools inference engine,” p. 7, Feb. 2023, doi: 10.1117/12.2667222.

[35] A. AlShehhi and R. Welsch, “Artificial intelligence for improving Nitrogen Dioxide forecasting of Abu Dhabi environment agency ground-based stations,” *J Big Data*, vol. 10, no. 1, Dec. 2023, doi: 10.1186/S40537-023-00754-Z.

[36] K. Yun *et al.*, “Development and validation of explainable machine-learning models for carotid atherosclerosis early screening,” *J Transl Med*, vol. 21, no. 1, Dec. 2023, doi: 10.1186/s12967-023-04093-8.

[37] K. Okarma and P. Lech, “A method supporting fault-tolerant optical text recognition from video sequences recorded with handheld cameras,” *Eng Appl Artif Intell*, vol. 123, Aug. 2023, doi: 10.1016/J.ENGAPPAI.2023.106330.

[38] MarschallOwen, ChoKyunghyun, and SavinCristina, “A unified framework of online learning algorithms for training recurrent neural networks,” *The Journal of Machine Learning Research*, vol. 21, pp. 1–34, Jan. 2020, doi: 10.5555/3455716.3455851.

[39] C. Wongwatkit, *A Development of Order Processing System: BPMN Model*.

[40] K. Meshkini, J. Platos, and H. Ghassemain, “An Analysis of Convolutional Neural Network for Fashion Images Classification (Fashion-MNIST),” *Advances in Intelligent Systems and Computing*, vol. 1156 AISC, pp. 85–95, 2020, doi: 10.1007/978-3-030-50097-9_10/COVER.

[41] “What is a REST API?” https://www.redhat.com/en/topics/api/what-is-a-rest-api (accessed Jun. 07, 2023).

[42] “What is JSON.” https://www.w3schools.com/whatis/whatis_json.asp (accessed Jun. 07, 2023).