






AKIP Process Automation Platform: A Framework for the Development of Process-Aware Web Applications

Ulisses Telemaco Neto¹^a, Toacy Oliveira²^b, Raquel Pillat²^c, Paulo Alencar¹,
Don Cowan¹^d and Glauca Melo¹^e

¹David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, Canada

²System Engineering and Computing Program, Federal University of Rio de Janeiro, Rio de Janeiro, Brazil

Keywords: Business Process Automation, Code Generation, Process-Aware Information System, BPMN.

Abstract: An increasing number of platforms for Business Process Automation (BPA) have been developed in recent years, including open-source and proprietary solutions. However, there are still some unsolved problems and limitations related to the adoption of these solutions, which include: vendor lock-in, limited UI/UX, limited integration, outdated technology stack, and lack of support for non-process features. The framework presented in this paper addresses these problems and limitations by providing an open-source platform to facilitate the development of *process-aware* information systems (PAISs) based on code generation techniques. The platform is capable of generating a functional *process-aware* Web application from a business process model defined in BPMN. To the best of our knowledge, there is no other software tool that generates fully functional *process-aware* Web applications. The presented framework has been evaluated in the academy and industry and used to develop dozens of non-profit and commercial process-aware applications.


1 INTRODUCTION


Despite the huge amount of capital and effort that has been invested in IT software solutions, the reality is that, for many organizations, including for-profit companies, most of their critical business processes are still being conducted without the support of automated, agile, modern, and user-friendly software systems. In many cases processes are conducted through email exchanges, phone calls, and manual integration between systems. This scenario may lead to several problems, including: *Vendor Lock-in*: One of the main risks of adopting a proprietary platform is being forced to continue using its product or service, regardless of quality, cost or any other reason, because it is not practical to switch away from that product or service. *Limited UI/UX*: Despite claiming that the systems provide features for building dynamic forms, the forms produced by the BPA platforms are still quite limited. In this way, the construction of com-


plex components (such as auto-complete input fields or elements whose validation rule is more sophisticated) becomes challenging and often unfeasible on some low-code platforms. *Limited Integration*: Many solutions for BPA provide extension mechanisms designed to integrate the platform with existing systems or third-party solutions. However, implementing such integrations are still costly, not efficient, and sometimes unfeasible. *Outdated Technology Stack*: Many of the existing BPA solutions are based on outdated techniques such as stateful (Dwyer, 2021) and server-side rendering (Iskandar et al., 2020) applications. *Lack of support for non-process features*: Another limitation of BPA platforms is that they focus only on process automation and often provide limited support for other features that are usually necessary for process execution. For example, customized authentication mechanisms, CRUD¹ for domain entities, and import/export data are typically features hard to develop in BPA solutions. There is a need of solutions that can support the automation of both process and Web-related features.


We have just mentioned a few problems that may happen in this scenario, but the list can be increased


¹Acronym for *Create, Read, Update, and Delete*

^a <https://orcid.org/0000-0002-7258-2623>

^b <https://orcid.org/0000-0001-8184-2442>

^c <https://orcid.org/0000-0002-5420-6966>

^d <https://orcid.org/0000-0002-5373-8522>

^e <https://orcid.org/0000-0003-0092-2171>

with many other issues such as process tracking, organizational agility, team communication and process redundancy. Because of the critical relevance of each of the items, it is difficult to identify the one that affects organizations the most. We see these pain points in almost every company that engaged with us and we believe that the solution involves the systematic automation of their processes. In this context, the adoption of Business Process Automation (BPA) techniques for building *Process-Aware* Information Systems (PAISs), i.e., systems whose requirements are described using process models such as BPMN, helps to address natively most of the problems just mentioned (Dumas et al., 2005).

There are several solutions for BPA, including open-source and proprietary platforms. These solutions will be briefly described in Section 3. The main problems and limitations of these solutions are:

1. *Vendor Lock-in*: One of the main risks of adopting a proprietary platform is being forced to continue using its product or service, regardless of quality, cost or any other reason, because it is not practical to switch away from that product or service.
2. *Limited UI/UX*: Despite claiming that the systems provide features for building dynamic forms, the forms produced by the BPA platforms are still quite limited. In this way, the construction of complex components (such as auto-complete input fields or elements whose validation rule is more sophisticated) becomes challenging and often not feasible on some low-code platforms.
3. *Limited Integration*: Many solutions for BPA provide extension mechanisms designed to integrate the platform with existing systems or third-party solutions. However, implementing such integrations are still costly, not efficient, and sometimes not feasible.
4. *Outdated Technology Stack*: Many of the existing BPA solutions are based on outdated techniques such as stateful (Dwyer, 2021) and server-side rendering (Iskandar et al., 2020) applications.
5. *Lack of support for non-process features*: Another limitation of current BPA platforms is that they focus only on process automation and often provide limited support for other features that are usually necessary for process execution. For example, customized authentication mechanisms, CRUD² for domain entities, and import/export data are typically features hard to develop in BPA solutions. There is a need for solutions that can

support the automation of both process and Web-related features.

To mitigate these problems and limitations, one approach is to develop a traditional web application and integrate it with a process engine. This way, the resulting software does not have the limitations of a platform focused on BPA and has the benefit of being controlled by a process engine. The difficult of manually integrating a web application with a process engine is that this integration is not straightforward and can be very challenging and costly (Samland and Turing, 2019) (Straube and Horn, 2021) (Shan, 2022).

In this context, our solution fills the gap by providing an open-source platform to facilitate the development of *process-aware* information systems based on code generation techniques. The platform is capable of generating a functional *process-aware* web application from a business process model defined in BPMN. To the best of our knowledge, there is no other software tool that generates fully functional *process-aware* web applications.

The framework was originally derived from an academic research agenda conducted by the AgileKIP Group³ that focuses on Knowledge Intensive Processes. In 2018, we released the first version of the platform and in 2019 we conducted the first POC (Proof of Concept) in industry. In 2020, we launched the second version of the platform that has been used to automate dozens of processes. The third version of the platform was just released and, among the new features are support for dashboards and modularization. The platform has been used in academia and industry for development of several non-profit and commercial process-aware applications. To support the demonstration of the proposed platform, we present in this paper a case study from industry encompassing the automation of three business processes in a process-oriented application.

The remainder of this paper is organized as follows: Section 2 presents the background of this research. The related work is briefly described in Section 3. The platform overview is presented in Section 4. Sections 5 and 6 discuss, respectively, how to use the platform and how to use a generated process-aware application. Technical details are discussed in Section 7 and a case study is presented in Section 8. Section 9 concludes the paper and presents future work.

²Acronym for *Create, Read, Update, and Delete*

³<https://agilekip.com/>

2 BACKGROUND

2.1 Business Process Model and Notation (BPMN)

BPMN as an ISO (ISO, 2013) and OMG (OMG, 2013) standard is leading technology for modeling business processes. Currently, BPMN is the business process notation most used in practice (Harmon, 2016) and with the greatest number of available tools. BPMN models can be interpreted and manipulated by both technical and non-technical personnel, reducing the likelihood of erroneous knowledge transfer (OMG, 2013). Moreover, BPMN can also express executable models that can be interpreted by process engines. In fact, systems such as Camunda, Flowable, and BonitaSoft deliver an integrated environment where users can design and execute BPMN models.

2.2 Business Process Automation (BPA)

Business Process Automation (BPA) is defined as the automation of complex business processes and functions beyond conventional data manipulation and record-keeping activities, usually through the use of advanced technologies (Kirchmer and Scheer, 2004). BPA is based on three pillars: orchestration, integration, and dynamic automated process execution (Mohapatra, 2009). Based on these pillars, BPA can be enabled by developing a systematic solution supporting a given business process.

2.3 Process-Aware Information System (PAIS)

A Process-Aware Information System (PAIS) is a particular type of application that uses information technology to manage and execute operational processes involving people, applications, and information sources (Dumas et al., 2005). PAIS requirements are described using process models such as BPMN, where activities, resources, decisions, events, and their relationships can be used to represent the flow of work.

2.4 Modern Process-Aware Information Systems

Modern PAISs have been built as process-aware web applications based on features that allow authorized users to manage (e.g., execute, view, query, delegate) their tasks and interact with the business processes

seamlessly. It is almost imperative that these applications provide rich user interfaces (light, fast, and user-friendly) on top of a reliable and scalable software architecture based on cutting-edge technologies, focused on cloud-native distribution, and easy integration with third-party applications.

3 RELATED WORK

Considerable research has been conducted in the domain of Business Process Management (BPM) and many tools have been proposed for different aspects related to Business Process Automation (BPA). Because of the wide range of solutions, it is a challenge to compare our platform with existing tools. However, we briefly describe a set of platforms for BPA that can be either a source of evaluation and/or inspiration for our framework. We divided those solutions into two groups: proprietary and open-source platforms.

The proprietary low-code solutions for BPA include *Aris*⁴ (Scholz and Wagner, 2004), *Bizagi*⁵, *Heflo*⁶, *Nintex*⁷, *Pipefy*⁸, *Process Street*⁹, *Signavio*¹⁰, and *SoftExpert BPM*¹¹. These solutions support BPMN or provide modeling tools based on BPMN that can be integrated with a low-code platform where users can create processes and business rules, add functional roles, create interfaces, customize forms, and manage related content in an integrated way.

Among the open-source platforms, we mention *Camunda*¹², *Bonita*¹³, *Flowable*¹⁴, and *JBPM*¹⁵. The *Camunda* Platform is an open-source workflow and decision automation platform. The solution is composed of tools that include a BPMN 2.0 process engine, a modeler, a cockpit, and a task-list manager. Together, these tools can be used for creating workflow and decision models, operating deployed models in production, and allowing users to execute workflow tasks assigned to them. *Bonita* is an open-source business process management and low-code development platform for BPA. The platform is composed of five

⁴<https://www.softwareag.com/>

⁵<https://www.bizagi.com/>

⁶<https://www.heflo.com/>

⁷<https://www.nintex.com/>

⁸<https://www.pipefy.com/>

⁹<https://www.process.st/>

¹⁰<https://www.signavio.com/>

¹¹<https://www.softexpert.com/>

¹²<https://www.camunda.org/>

¹³<https://www.bonitasoft.com/>

¹⁴<https://www.flowable.com/>

¹⁵<https://www.jbpm.org/>

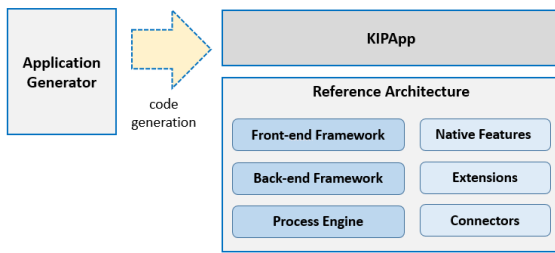


Figure 1: Platform overview.

main components: *Bonita Studio*, *Bonita BPM Engine*, *Bonita Portal*, *Bonita UI Designer*, and *Bonita Continuous Delivery*. *Flowable* is a light-weight business process engine written in Java. The platform supports the deployment of BPMN 2.0 and can be used for creating process instances of those process definitions, running queries, accessing active or historical process instances and related data. *jBPM* (Java Business Process Model) is an open-source workflow engine written in Java that can execute business processes described in BPMN 2.0 (or in jPDL, its own process definition language, in earlier versions). *jBPM* is a toolkit for building business applications to assist in automating business processes and decisions. It is maintained by Red Hat Inc., part of the JBoss community and closely related to the systems Drools and OptaPlanner projects in the KIE group. It is released under the ASL (or LGPL in earlier versions) by the JBoss company.

The limitations of these existing solutions were described in Section 1, and, as a reminder, are listed here as well: (1) *Vendor Lock-in*; (2) *Limited UI/UX*; (3) *Limited integration*; (4) *Outdated technology stack*; and (5) *Lack of support for non-process features*.

4 PLATFORM OVERVIEW

AKIP Process Automation Platform is an open-source project devoted to facilitate process automation initiatives based on code generation techniques. The platform was built by developers and researchers aiming to disseminate the use and development of process-aware information systems, more specifically modern web applications that we call *KIPApps*. Figure 1 presents an overview of the platform. It consists of two components supporting *KIPApp* development: an *Application Generator* and a *Reference Architecture*.

4.1 KIPApp

In our solution, we call *KIPApp* (which stands for *Knowledge Intensive Process Applications*) a modern process-aware web application (such as described in Section 2.4) that executes on the top of an open-source process engine. In short, a *KIPApp* provides the following main features:

- **Web Application Management:** This includes the management of users, configuration properties, health checks, logs, and application metrics.
- **Process Management:** It includes the management of domain entities, deployed processes, tenants, and process instances.
- **Task List:** It provides an updated user tasks list showing tasks completed, assigned, and waiting to execute.

4.2 Application Generator

The application generator is a tool to generate, develop, and deploy *KIPApps* quickly. It generates code for the base reference architecture of the AKIP platform.

The generator is a key tool in our solution because it accelerates the development process by scaffolding a huge amount of code that, without the generator, would have to be coded manually.

4.3 Reference Architecture

The reference architecture represents the backbone of a *KIPApp*. It is composed of front and back-end frameworks, a process engine, native features, extensions, and connectors. Details on technologies used for front and back-end frameworks and the process engine are presented in Section 7. The process engine is the tool used in our solution to orchestrate the process workflow from an executable BPMN model. Next, we depict the remaining three elements of the reference architecture:

- **Native Features** are common features already provided by the reference architecture, meaning there is no need to repeat the code of these features in each *KIPApp*. Examples of native features include *Advanced Tasks List*, *Start-Process*, *Process-Instances Dashboard*, *Management of Deployed Processes*, and *User Management*.
- **Extensions** are components that allow seamless integration of the *KIPApp* with the process engine.



Figure 2: Main steps to use the platform.

- **Connectors** are components that allow the application to integrate quickly with the external world. Examples of connectors include an *Email Connector* used to send email automatically throughout the process execution, *RestAPI Connector* to integrate with other systems through Rest APIs, *JMS Connector* to carry out communication through messaging, and *AWS Connectors* that allow integration with main AWS services. The main idea behind these elements is to reuse the most common integration components through a sophisticated and highly configurable set of connectors.

5 PLATFORM WALK-THROUGH

Using the platform involves three main steps (Figure 2):

1. Generation of a KIPApp based on the Reference Architecture.
2. Technical implementation of one or more business processes.
3. Deployment of the business process(es) into the KIPApp.

We briefly explain below how these general steps can be performed. Details on how to install and use the platform can be found in the Github project public repository.¹⁶

5.1 Generating a KIPApp

Our application generator was designed so that the user only needs to execute a single command and answer some questions related to the application configuration from a wizard to get your web application running with several features already installed. For more details, check out the platform’s Github public documentation.

5.2 Implementing a Business Process

In this section, we will focus on the development of an illustrative process named *Travel Plan Process*. In



Figure 3: *Travel Plan Process* - a running example.

a nutshell, this process has steps that users execute when planning a trip. Therefore, our goal is to develop a process-aware web application that aids a user planning a trip. For illustrative purposes, we will use a very simple version of this process, composed of only three user tasks, as shown in Figure 3:

To implement this business process, we should perform the following steps:

1. Define the business process model;
2. Define the domain model;
3. Generate domain entities;
4. Generate process entities; and
5. Customize the generated code.

5.2.1 Defining the Business Process Model

The business process model should be specified in BPMN¹⁷ and contain only typical tasks (usually User, Service, or Message Tasks) as illustrated by the process model in Figure 3. Moreover, the process should have an associated identifier (id) and be defined as executable. Optionally, we can also specify a description for the process using markdown notation. This description will be shown in the application when a new process instance is initiated.

5.2.2 Defining the Domain Model

A domain model (e.g., in UML notation) should be created in order to help identify domain entities that the business process (supported by the KIPApp) needs to handle. To keep our running example simple, the domain model is represented by a single entity named *TravelPlan* that has a few properties as shown in Figure 4. This entity represents the data that are handled by the process.

It will serve as a guide artifact for the next steps (related to entity generation). However, the domain model itself is not an input artifact for the AKIP platform. In fact, it uses domain specifications in JSON (JavaScript Object Notation) format, which is a standard data interchange format for the Web. JSON files are lightweight, text-based, human-readable, and can be edited using a text editor. Figure 5 presents the metadata of the *TravelPlan* domain entity in JSON format. The `fields` element describes primitive types of the entity and the `relationships` element

¹⁶<https://agilekip.github.io/pap-documentation>

¹⁷For now, only Camunda Modeler is supported.



Figure 4: Simplified domain model.

```

{
  "fields": [
    { "fieldName": "name", "fieldType": "String" },
    { "fieldName": "startDate", "fieldType": "LocalDate" },
    { "fieldName": "endDate", "fieldType": "LocalDate" },
    { "fieldName": "airlineCompanyName", "fieldType": "String" },
    { "fieldName": "airlineTicketNumber", "fieldType": "String" },
    { "fieldName": "hotelName", "fieldType": "String" },
    { "fieldName": "hotelBookingNumber", "fieldType": "String" },
    { "fieldName": "carCompanyName", "fieldType": "String" },
    { "fieldName": "carBookingNumber", "fieldType": "String" }
  ],
  "relationships": [],
  "entityType": "domain",
  "service": "serviceClass",
  "dto": "mapstruct",
  "jpaMetamodelFiltering": false,
  "readOnly": true,
  "pagination": "no",
  "name": "TravelPlan",
  "skipFakeData": true
}
    
```

Figure 5: *TravelPlan* domain entity metadata.

should describe its relationships (when they exist) with other domain entities.

```

"entityType": "process-binding",
"processBpmnId": "TravelPlanProcess",
"domainEntityName": "TravelPlan",
"name": "TravelPlanProcess",
    
```

Figure 6: Fragment of the *TravelPlanProcess* process-binding entity metadata.

```

{
  "fields": [
    { "fieldName": "name", "fieldType": "String" },
    { "fieldName": "startDate", "fieldType": "LocalDate" },
    { "fieldName": "endDate", "fieldType": "LocalDate" }
  ],
  "relationships": [],
  "entityType": "start-form",
  "processBpmnId": "TravelPlanProcess",
  "processEntityName": "TravelPlanProcess",
  "domainEntityName": "TravelPlan",
  "service": "serviceClass",
  "dto": "mapstruct",
  "jpaMetamodelFiltering": false,
  "readOnly": false,
  "pagination": "no",
  "name": "TravelPlanStartForm",
  "skipFakeData": true
}
    
```

Figure 7: *TravelPlanStartForm* start-form entity metadata.

Figure 8: KIPApp’s UI form used to start a *Travel Plan* process instance.

5.2.3 Generating Domain Entities

Once we have specified the *TravelPlan* domain entity in a JSON file (such as illustrated in Figure 5), we can generate all the files supporting the manipulation of this entity in the Web application (KIPApp). The “entity” sub-generator of the AKIP platform will create all the necessary application files (including the database table, the controller of basic CRUD opera-

tions, and services) and provide a CRUD front-end for each generated entity. The CRUD support, however, will only be provided if the `readOnly` property of the JSON entity specification is set to `false`. In the case of our running example, as the *TravelPlan* domain entity is read-only (see Figure 5), it will only be possible to view its instances (created from the *TravelPlanProcess*). To view or handle instances of domain entities, we need to use the *Entities* application menu.

5.2.4 Generating Process Entities

In this step, we need to generate the KIPApp's entities related to the process support. Each process entity type plays a different role in the application:

- **Process-binding Entity:** This entity represents the binding between a business process (BPMN model) and its corresponding domain entity. In other words, the aim of this entity is to indicate the domain entity associated with the process.
- **Start-form Entity:** This entity is used to generate the User Interface (UI) form which starts a process instance.
- **User-task Entity** This entity is used to generate the UI form for a specific user task present in the process.

As in the case of domain entities, process entities also need to be specified as JSON files. In our example, the **process-binding entity** is named *TravelPlanProcess*. Its JSON specification is very similar to the *TravelPlan* domain entity presented in Figure 5. For this reason, we will not show here its complete specification, but only some of its details. *TravelPlanProcess* differs from the *TravelPlan* entity only in relation to the properties presented in Figure 6. Specifically, we highlight the `processBpmnId` and `domainEntityName` properties. The first one should match the process Id defined in the BPMN process model (as mentioned in Section 5.2.1). The second one (`domainEntityName`) should match the domain entity previously created (i.e., *TravelPlan*).

Concerning the **start-form entity**, Figure 7 presents its JSON specification for our running example. The `fields` element describes the fields from the domain entity that should be present in the process start UI form whereas the `relationships` element describes the relationships from the domain entity that should be included in this form. The `entityType` property is *start-form*; `processBpmnId` matches the process id defined in the BPMN model; `processEntityName` matches the process-binding entity previously created; and `domainEntityName` matches the domain entity previously created. Figure 8 shows the KIPApp's UI form generated from

the *TravelPlanStartForm* entity (Figure 7) by using the "entity" sub-generator of the AKIP platform. This UI form will be used to start a Travel Plan process instance in the Web application.

A **user-task entity** should be created for each user task present in the business process model. In the case of our example, we need to create three user-task entities (named *TaskFlight*, *TaskHotel*, and *TaskCar*), because the Travel Plan process contains three user tasks (see Figure 3). Owing to space limitations, we will not present here the JSON specifications for these entities, but they can be found in the platform's online tutorial. However, they are very similar to the start-form entity specification previously created (Figure 7), as both entity types (user-task and start-form) are used to generate UI forms of the resulting KIPApp. In a user-task entity, the `entityType` property should be *user-task-form* and the `fields` element should contain the fields from the domain entity that should appear in the KIPApp's UI form used to execute the process's user task. Additionally, it should contain the `taskBpmnId` property matching the corresponding task id defined in the BPMN model file.

All entities mentioned above (i.e., the domain and process ones) can be generated at the same time to build the KIPApp by using the "entity" sub-generator of the AKIP platform. This generator will create all the necessary application files to support such entities.

5.2.5 Customizing the Generated Code

In a real scenario, once the application code has been generated, it usually still needs some customizations performed by developers. The most common customizations we have identified in practice are the addition of support for business rules and adjustments in UI form fields.

5.3 Deploying a Business Process into a KIPApp

Finally, the KIPApp is ready and all entities supporting the execution of the *Travel Plan* business process were generated and customized. Next, we need to explicitly deploy this process into the KIPApp.

To do that, it is necessary to log into the application using an account with *admin* privileges, access the *Process Definitions Management* feature (Figure 9), click on the *Deploy a Process* button, and choose the business process model file (BPMN) corresponding to the *Travel Plan* Process (detailed in Section 5.2.1). The bottom of Figure 9 shows the *Travel Plan* process already deployed.

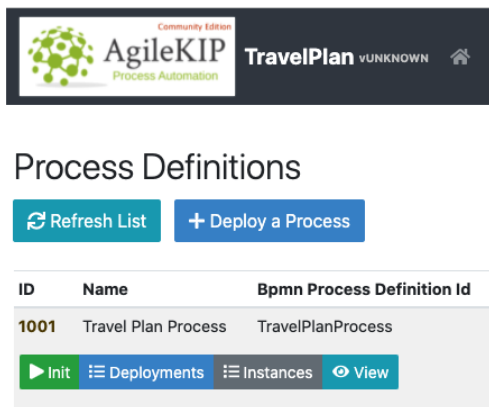


Figure 9: KIPApp screen showing a deployed business process.

6 USING A KIPApp

This section briefly presents how to use a KIPApp to execute a business process. Owing to space limitations, we will not present other functions provided by the application.

Enacting a business process involves at least (1) starting a new process instance, and (2) executing user tasks from this instance. The following subsections describe how these activities can be performed within a KipApp.

6.1 Starting a Process Instance

Once the *Travel Plan* process has been deployed, we can start the process by clicking on the *Init* button from the app screen shown in Figure 9. The start form of this process (shown in Figure 8) contains the fields we defined previously in the *TravelPlanStartForm* entity metadata (Figure 7). The process description (in this case, three enumerated steps) came from the *documentation* element of the BPMN process model. The process instance will actually start when we click on the *Start* button at the bottom of the form (Figure 8). To see the instances of the process already created, we click on the *Instances* button from the screen in Figure 9.

6.2 Executing User Tasks from a Process Instance

A KIPApp user can see the updated process tasks list from the *My Tasks* app menu. In the case of the *Travel Plan* process, its tasks are available for any authenticated user. Figure 10 shows the user tasks screen where one can note the *Choose flight* task has already

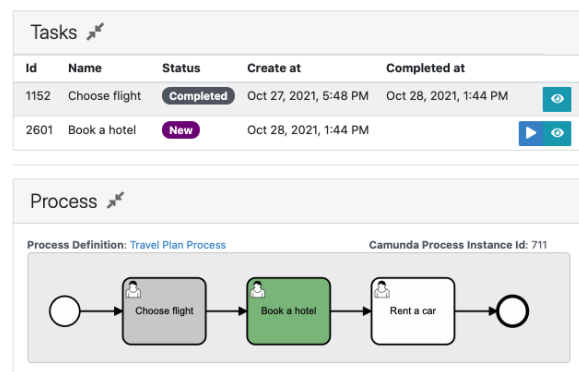


Figure 10: User tasks from a process instance.

been completed (owing to its status) and the *Book a hotel* task is enabled for execution (the BPMN process model in this screen shows in green the current task waiting to be executed). To execute it, we only need to click on the *Play* button on the right side of the task. A process instance will be concluded after all its tasks have completed.

7 TECHNICAL NOTES

This section provides some information about the platform's implementation concerning its support technologies. A KIPApp is a single web page application generated by the AKIP platform, running on top of a process engine. Our Application Generator is an extension of the *JHipster*¹⁸ generator (technically it is a JHipster blueprint) that overrides many sub-generators and provides its own templates and functions. Our Reference Architecture is based on one of the most modern tool stacks in the software industry. Currently, we use *VueJS*¹⁹ as a front-end framework, *Spring Boot*²⁰ as a back-end framework, and *Camunda Platform* as a process engine.

8 CASE STUDY

The AKIP platform is used in practice within a software development company as a support tool to build process-aware web applications (KIPApps). This section presents data on a case study conducted in the context of this enterprise. The following subsections present the goal of our study (8.1), data about the company and the automated business processes (8.2),

¹⁸<https://www.jhipster.tech/>

¹⁹<https://www.vuejs.org/>

²⁰<https://spring.io/projects/spring-boot>

the procedure that was followed (8.3), the results of the case study (8.4) and, finally, some considerations about the case study (8.5).

8.1 Goal

Our goal with this study was to assess the feasibility and usability of the AKIP platform. In other words, we intended to verify its effectiveness to generate process-aware web applications in practice in the context of a software development company. Specifically, the study aimed to answer the following research question (RQ):

RQ: Is the AKIP platform able to generate process-aware web applications (KIPApps) from real-world business process models?

8.2 Subject Data

This case study was conducted in the context of a Brazilian software development company (with approximately 30 employees), located in the city of Rio de Janeiro. Its focus is on the development of IT solutions for the logistics and port sectors.

The case study was conducted using one of the company's projects, named the *Process Automation* project, which is automating and integrating business processes from a client company by providing a customized process-aware web application. The project has a team composed of a project manager, a product owner, a process analyst, a software analyst, and two developers. The project follows a Scrum-based agile development process that delivers an automated business process at each iteration (sprint).

We evaluated three business processes automated in this project, which are from the logistics and sea-port domains. However, as these processes are proprietary and confidential, we will only present general information about their models, without presenting any details of the process model itself.

Table 1 shows the size of the BPMN process models automatically generated with the AKIP platform (identified by P1, P2, and P3) in terms of number of tasks, subprocesses, gateways, events and lanes, and provides information about the complexity of these models.

8.3 Procedure

First of all, the software architect generated the KIPApp for the client company using the AKIP platform. Afterwards, at each project's development process iteration (sprint), the team followed the sequence

Table 1: Data on case study business process models.

<i>Process Model Data</i>	<i>P1</i>	<i>P2</i>	<i>P3</i>
Lanes	5	4	2
Subprocesses	0	1	0
Tasks	34	29	15
User tasks	6	6	4
Message tasks	15	15	6
Service tasks	13	8	5
Gateways	11	10	2
Parallel gateways	0	0	0
Exclusive gateways	11	10	2
Inclusive gateways	0	0	0
Events	7	7	4
Start events	1	1	1
End events	3	5	2
Conditional events	3	1	1

of steps presented in Section 5.2 for implementing a business process:

1. The BPMN business process model was built by the process analyst, interacting with the product owner and users of the process.
2. The domain model was created by the software analyst with the participation of the process analyst.
3. The domain and process entities metadata were specified by the developers in JSON and used by the AKIP platform that generated the code base for the given process.
4. The generated app's code was customized by developers (including support for business rules and integrations) and UI forms (screens) were customized by the software analyst.
5. Finally, the business process was deployed in the KIPApp and the application was available for client validation.

8.4 Results

As shown in Table 1, business processes considered in this case study are composed of gateways, events, and different types of BPMN tasks (User, Message, and Service tasks) that allow us to evaluate advanced code-generation resources of the AKIP platform. We could not present such advanced resources in this paper owing to space limitations, but they are covered in the platform's online tutorial. Moreover, all business processes considered in this study could be successfully automated through the generated KIPApp. Code customizations are always necessary, but even so, the platform contributes to the automatic generation of a large volume of application code, which would have

to be developed manually without the platform. The project team where the platform has been used reported to us that it is very useful in streamlining the work of application development.

Finally, we conclude this section by answering the research question that motivated this case study: *Is the AKIP platform able to generate process-aware web applications (KIPApps) from real-world business process models?* The evaluation that was conducted allowed us to conclude that the AKIP platform was effective in generating a Web application supporting the business processes implemented in the context of the evaluated company's project. Based on the cases we evaluated in this practical study, we can claim that the AKIP platform satisfactorily generates useful and modern application code supporting business process execution. However, more case studies should be analyzed to extend the platform's evaluation and consolidate its validation.

8.5 Discussion

The AKIP Process Automation Platform has been used in several realistic and academic scenarios by different teams to support developing process-aware Web applications, and in this paper we present a case study to demonstrate the feasibility of the platform. In future work, we intend to conduct a survey with members of the project's development team in order to evaluate specific aspects about the use of the AKIP Process Automation Platform.

It is important to mention that the business processes considered in this evaluation come from a single enterprise's project and may not be representative of those occurring in other realistic settings. Different software development processes, domains and organizations may lead to different results. Thus, our platform should be tested further on business process automation scenarios from other organizations and domains.

As can be seen in Table 1, the process models considered in the case study include only one type of BPMN gateway (exclusive gateway) and one type of non-essential event (conditional event). Thus, the effectiveness of the AKIP platform in automating processes with different types of BPMN gateways is not evaluated in this study. Other BPMN process elements have not yet been tested using the platform, such as transactions, compensation activities, and business rule tasks. In future work, we intend to address this limitation.

In addition, we want to point out that the case study (as well as our experience automating business processes with the platform) shows the AKIP Process

Automation Platform can be used successfully in conjunction with an agile software development process such as Scrum to aim the development of *process-aware* information systems. The project team that participated in the case study reported that the platform helps them to achieve more agility in implementing business processes.

In summary, our results are promising, as demonstrated by several independent uses, and we are still working to improve the platform in order to support more BPMN elements and include features directed toward the low-code movement.

9 CONCLUSIONS AND FUTURE WORK

In this paper we have introduced the AKIP Process Automation Platform, an open-source project devoted to facilitate business process automation initiatives based on code generation techniques. The platform is capable of generating a functional process-aware Web application, which we call KIPApp, from an executable business process model defined in BPMN. To the best of our knowledge, there is no other software tool that generates fully functional *process-aware* Web applications. Our solution is different in that the generated application runs on top of a process engine responsible for orchestrating the execution of business processes and the generated Web application's architecture is based on modern technologies currently used by the software industry.

In addition, we have presented a real-world case study in which the AKIP platform has been evaluated. The results of this study showed that it was effective in generating a Web application supporting the business processes implemented in the evaluated study. In future work, we intend to conduct other case studies, evaluating the tool in new business domains, in order to broaden its validation.

We are also working on the development of a new component for the AKIP platform. This component consists of a software development process based on agile practices, which we call SCRUB4PA (SCRUB for Process Automation), and defines in detail how a KIPApp should be built. This process will encompass phases, tasks, roles, artifacts, templates, and tools used throughout the development of a KIPApp.

REFERENCES

- Dumas, M., Van der Aalst, W. M., and Ter Hofstede, A. H. (2005). *Process-aware information systems: bridging*

- people and software through process technology*. John Wiley & Sons.
- Dwyer, G. (2021). Stateful vs stateless architecture: Why stateless won. <https://www.virtasant.com/blog/stateful-vs-stateless-architecture-why-stateless-won>.
- Harmon, P. (2016). The state of business process management 2016. Technical report.
- Iskandar, T. F., Lubis, M., Kusumasari, T. F., and Lubis, A. R. (2020). Comparison between client-side and server-side rendering in the web development. In *IOP Conference Series: Materials Science and Engineering*, volume 801, page 012136. IOP Publishing.
- ISO (2013). ISO/IEC 19510:2013: Information technology – Object Management Group Business Process Model and Notation. Technical report, Organization for Standardization.
- Kirchmer, M. and Scheer, A.-W. (2004). Business process automation—combining best and next practices. In *Business Process Automation*, pages 1–15. Springer.
- Mohapatra, S. (2009). *Business process automation*. PHI Learning Pvt. Ltd.
- OMG (2013). Business Process Model and Notation (BPMN), Version 2.0.2. Technical report, Object Management Group.
- Samland, F. and Tuting, W. (2019). Monolith to microservice, waterfall to agile, success with camunda. <https://camunda.com/customer/deutsche-telekom/>.
- Scholz, T. and Wagner, K. (2004). Aris process platform tm and sap netweaver tm: Next generation business process management. In *Business Process Automation*, pages 29–37. Springer.
- Shan, D. (2022). Finance back-office billing engine using camunda. <https://camunda.com/customer/atlassian/>.
- Straube, C. and Horn, D. (2021). Scaling process automation with a modular open source platform. <https://camunda.com/customer/city-of-munich/>.