


RESEARCH ARTICLE

WILEY

Analyzing business process management capabilities of low-code development platforms

Apurvanand Sahay¹  | Davide Di Ruscio¹ | Ludovico Iovino² | Alfonso Pierantonio¹

¹Department of Engineering and Information Sciences and Mathematics, University of L'Aquila, L'Aquila, Italy

²Gran Sasso Science Institute, L'Aquila, Italy

Correspondence

Apurvanand Sahay, Department of Engineering and Information Sciences and Mathematics, University of L'Aquila, L'Aquila, Italy.

Email: apurvanand.sahay@univaq.it

Funding information

Lowcomote Training Network; H2020 Marie Skłodowska-Curie Actions, Grant/Award Number: 813884

Abstract

Low-code development platforms (LCDPs) aim to simplify software systems' development by providing easy-to-use graphical interfaces and drag-and-drop facilities. The system behaviors are defined through available data handling and workflow mechanisms enabling the specification of business processes from users that do not have strong programming skills. However, the number of LCDPs has grown significantly over the last few years. Consequently, it is not easy for inexperienced users to understand their differences, especially in terms of provided modeling constructs. In this article, we analyze and compare eight low-code development platforms by focusing on their capabilities for specifying business processes. The analysis exploits business process modeling and notation (BPMN) as a reference modeling language. Thus, the core elements of BPMN are leveraged to analyze the workflow mechanisms provided by each of the analyzed LCDP. The article explains different types of process flows and data handling means of the different LCDPs aiming to give potential users objective elements that can be used to make educated decisions when selecting LCDPs.

KEYWORDS

business process management, low-code development platforms, model-driven engineering

1 | INTRODUCTION

Low-code development platforms (LCDPs)* provide development environments to conceive software applications through graphical interfaces instead of traditional hand-coded programming techniques.¹ LCDPs offer workflow modeling notations to define business processes and automate the corresponding tasks, including data change, scheduling, or connecting information flows to external services. For example, when an employee is added to a company, an automated welcome email is sent to give some essential information. Such automation helps avoid manual and repetitive work and can produce a better and more efficient result in the overall business process of that company.

The use of LCDPs to automate business processes aims at reducing the cost and time of implementing and maintaining software systems by satisfying different groups of technical and non-technical users, for example, software developers and business analysts, respectively.² Furthermore, such business process automation helps develop various applications such as event monitoring, process automation, approval process control, escalation management, inventory management, quality management, and workflow management.³ As discussed in Reference 4, business process management aims to

Abbreviations: BPMN, business process modeling and notation; LCDPs, low-code development platforms.

*Hereafter, the terms *low-code platforms* and *low-code development platforms* are used interchangeably and are abbreviated as LCDPs.

This work is equally contributed by all authors.

support three main activities: (i) modeling business processes to capture different workflow specifications, (ii) optimizing business processes, and (iii) automate the workflow to generate its implementation. LCDPs provide modeling constructs to specify workflows and handle corresponding data in this respect. One of the critical use cases of LCDPs is to automate business processes, workflows, and case management that involves complex business logic that can operate with external services and multiple end-user roles.⁵ However, selecting the right LCDP that can be employed to specify, automate, and manage the business processes at hand is a difficult task, which is further complicated by the increasing number of LCDPs being continuously released.

In this article, we analyze and compare eight low-code development platforms by focusing on their capabilities for specifying and managing business processes. The considered definition of business processes is based on the standard business process modeling and notations (BPMN).⁶ BPMN plays the role of the reference modeling language to support the comparison and the analysis of the considered LCDPs.

The main contributions of the article are as follows.

- We analyze the workflow and data handling mechanisms of different LCDPs associated with their respective process modeling constructs.
- We identify the existing and missing mapping between components in BPMN-like modeling language and the analyzed low-code platforms.
- We discuss a set of quality criteria inspired by BPMN that can be used to evaluate modeling constructs of different LCDPs.

Structure of the article. Section 2 describes the core elements of BPMN used for modeling business processes by introducing an explanatory and running example. Related works for analyzing various process modeling platforms are discussed in Section 3. Section 4 illustrates various LCDPs related to BPMN modeling, by relying on the introduced running example. Section 5 derives aspects that must be discussed in a comparison between BPMN and LCDPs. Section 6 exploits BPMN quality criteria to pave the way toward a possible quality evaluation of LCDPs. Section 7 concludes the article and draw some perspective future work.

2 | PROCESS MODELING WITH BPMN

Process modeling is widely used within organizations as an approach to describe through graphical notations how business entities conduct their operations⁷ in terms of activities, events, and control flows. In addition, process models may also represent information about involved data, resources, and other related artifacts. Petri nets⁸ and BPMN[†] are two prominent examples of process modeling techniques. BPMN was developed by Object Management Group[‡] (OMG) and was released for the first time in May 2004. The BPMN specification was created to provide a notation intended to be “*understandable by all business users, from the business analysts who create the initial drafts of the processes, to the technical developers responsible for implementing the technology that will perform those processes, and, finally, to the business people who will manage and monitor those processes.*”⁹

In BPMN, business processes are generally called workflows, and they are specified in terms of a dedicated notation consisting of various modeling constructs for activities, events, gateways and so forth. In addition, BPMN also includes composition mechanisms, for example, orchestrations, endowed with corresponding execution semantics. The available constructs help technical and non-technical individuals to understand their views of the business model by hiding or revealing the necessary detail according to the role, designation, or expertise of the person involved. The execution of BPMN specifications is typically performed by transforming models to an XML-based business process execution language (BPEL) that helps to optimize and automate the defined business processes.¹⁰ BPMN 2.0,⁶ released in 2011, expands earlier BPMN 1.x capabilities adding choreography diagrams for modeling several business interactions. Overall, BPMN consists of notations and semantics of three different diagrams, that is, *collaboration*, *process*, and *choreography* diagrams. In this article, we focus on the core BPMN elements for modeling business processes as shown in Table 1.[§]

[†]<https://www.omg.org/spec/BPMN/2.0/>

[‡]<https://www.omg.org/>

[§]Some of the considered BPMN core elements are taken from https://www.omg.org/bpmn/Samples/Elements/Core_BPMN_Elements.htm.

TABLE 1 BPMN core elements

Core elements	Short description
<i>Swimlanes</i>	
Pool	Defines a process participants or an external process. Can have multiple lanes.
Lane	Define a specific participant or role within a process inside a pool.
<i>Flow object</i>	
Event	Shows something that has happened or may about the happen during a process.
Activity	Work performed within a process by a participant.
Gateway	Controls the sequence & flow within a process, providing routing within a process.
<i>Data</i>	
Data object	Information required or produced by an activity that persists only during a process.
Data store	Information required or produced by an activity that persists beyond the process.
Message	Contains the information between two participants (usually across pools).
<i>Artifacts</i>	
Group	Allow flow objects to be grouped for documentation and analysis.
Annotation	Provides additional information on an element within a process.
<i>Connecting objects</i>	
Sequence flow	The connection between flow object in a process shows execution within a process.
Message flow	Shows message between process participants (usually across pools).
Association	Links data/artifacts to flow object shows the direction.

In the following, we describe an illustrative and running example of a *Recruitment Drive* process. The applicant reads the job notification on the company's website and checks her eligibility criteria. If ineligible, she will close the website; otherwise, she clicks on the application link given on the notification. She further fills out the application and submits it. This will notify the Human Resource (HR) manager, who checks and validates the application. If the application is incorrect, the application will be rejected, and thus the application form gets closed. Alternatively, if the application is correct, an interview is set up. Finally, the manager sends an email to the applicant notifying the time and place for the interview. Consequently, the manager also conducts interviews in which the applicant participates. After the interview, the manager sends the pass or fail message to the applicant. If the applicant passes the interview, the manager sends the applicant's pass message to the administrative department. Upon receiving the notice from HR, the administrative department issues a registration sub-process to complete the applicant's hiring.

Figure 1 depicts a fragment of the BPMN2.0 specification of the *Recruitment Drive* example, which has been modeled employing core elements shown in Table 1. There are two *lanes* namely Company A and Applicant. The former contains two *pools* namely HR and Administration. The applicant sends the application *message* to notify HR. The HR sends back the interview intimation to the applicant followed by pass or fail *message* of the interview based on the fail decision making *gateway*. Also, HR sends a pass *message* to the admin based on the pass decision making *gateway* and the admin starts the registration sub-process *activity*. The *flow object* from one *event* to an *activity* is also shown multiple times in Figure 1. For instance, the start *event* "Check recruitment notification from company's website" to the *activity* "Check eligibility criteria" is attached via a *flow object*. Also, there are multiple *message flow* elements connecting different send and receive *messages* that run across *pools* or *lanes*.

3 | RELATED WORK

This section discusses works that analyze process modeling languages and compare them based on relevant criteria. The purpose of this section is to give a mini systematic literature review by following consolidated guidelines for this kind of works in the area of software engineering.¹¹ To identify and discuss current results, we observed a process consisting

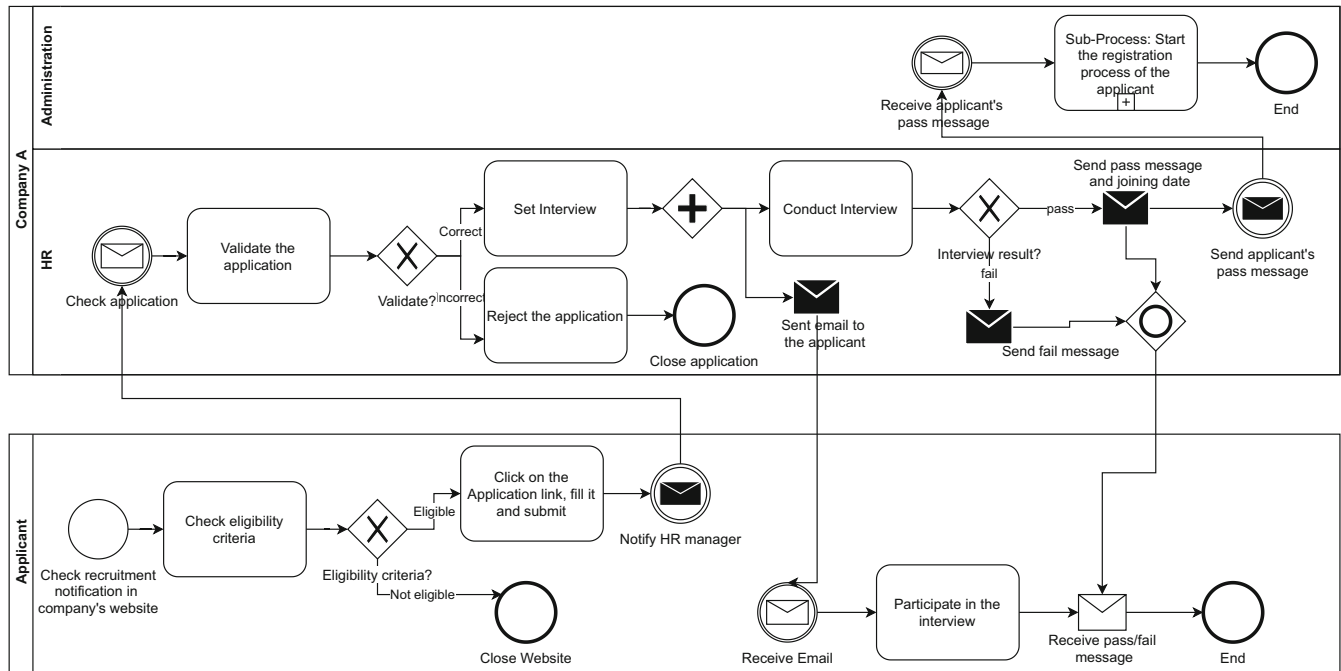


FIGURE 1 Fragment of the BPMN2.0 specification of the *Recruitment Drive* example

of the following steps: (i) definition and execution of a query string to automatically collect papers of interest based on a set of keywords; (ii) manual filtering the retrieved papers with respect to defined inclusion and exclusion criteria; (iii) discussion of the remaining papers by answering the following research question:

RQ: What are the key aspects that are considered when comparing business process modeling languages?

Details of the applied methodology are given in Section 3.1. The obtained results are presented in Section 3.2, whereas the answer to the defined research question is given in Section 3.3.

3.1 | Study design

As previously mentioned, we defined a query string to be executed on different digital resources to collect papers that analyze existing work comparing business process modeling languages. To this end, the *query string* that we conceived is the following:

“BPMN” OR Business “Process Modeling Language”) AND ((“Qualitative Comparison” OR “Analysis”) AND (“Business Process Modeling Languages” OR “BPMN”))

It has been applied on the following *electronic resources*:

1. *IEEE Xplore*[¶];
2. *Science Direct*[#];
3. *Association for Computing Machinery (ACM) Digital Library*^{||};
4. *Springer Link*^{**};

[¶]<https://ieeexplore.ieee.org/Xplore/home.jsp>

[#]<https://www.sciencedirect.com/>

^{||}<https://dl.acm.org/>

^{**}https://link.springer.com/chapter/10.1007/978-3-319-31232-3_58

5. *Academia*^{††};
6. *Scientific Research*^{‡‡}.

We decided the selection criteria of this study before the query definition phase to reduce the likelihood of bias. In the following, we describe the inclusion (I) and exclusion (E) criteria of our study:

- I1 Papers that have been published in the period 2000–2022;
- I2 Studies subject to peer review (e.g., papers published as part of conference proceedings or journal publications are considered, whereas white papers are discarded);
- I3 Studies that qualitatively compare different modeling languages and are available for consultation;
- E1 Papers that are not written in English;
- E2 Short papers of tutorial slides;
- E3 Book chapters.

Applying such inclusion and exclusion criteria to the papers initially collected by the query execution, we obtained 10 documents that are discussed in the next section.

3.2 | Results

In Reference 12, the authors discuss the software process modeling language (SPML) and compare it with the UML-based SPML. The article focuses on the framework that involves setting up the requirements for process modeling: semantic richness (expressiveness), modularity, executability, conformity to UML-standard, and formality. This article also evaluates the suitability of the known UML-based SPMLs for process modeling.

The paper co-authored by Bendraou et al.¹² focuses on the criteria that distinguish different SPMLs like semantic richness and expressiveness. These aspects include core process elements such as activity, artifact, role, and agent. One of the other aspects of expressiveness is activities, and action coordination that involves proactive control and reactive control.¹³ Proactive controls specify the order in which the activities are executed. They are precedence relationships (i.e., start-start, start-finish, finish-start, and finish-finish) and call actions such as explicitly calling an activity, operation and so forth. Following proactive control, reactive control specifies conditions or events in response to the executed activities. Reactive control includes exceptions, event handling and so forth. Also, exception handling is one of the criteria for expressiveness that explains the recoverability of a stable state of the process that handles failure.

Garcia et al.¹⁴ describe a quality model (QM) used in a framework called quality evaluation framework (QuEF) that manages the quality requirements of a modeling environment. The QM formulates different approaches that characterize the quality of model-based SPML. These approaches include expressiveness, understandability, conformity to standards, granularity, executability and orchestability, measurability, business rules, supporting tools, and validation in real environments. With the help of these approaches, ten representative SPMLs are evaluated and compared. These SPMLs are of three types: metamodel-level approaches, SPML based on UML, and methods based on standards. Lastly, this article has many model-based proposals for SPML. Furthermore, this article has taken newer aspects to characterize SPMLs. Such characteristics are validation within enterprise environments, friendly support tools, mechanisms to carry out continuous improvements, mechanisms to establish business rules, and elements for software process orchestration. All the quality criteria mentioned in this article have been described in Section ? concerning various LCDPs and compared with the BPMN modeling standard.

In Reference 15, the authors analyze how SPEM (Software Process Engineering Metamodel Specification) and BPMN (Business Process Modeling Notation) standards are represented in a software process modeling context. This analysis is done by defining a standard structure based on a process ontology to define a software process. Then the SPEM and BPMN standards are semantically matched with the default modeling process. These notations are also mapped with components of QMs named CMMI-DEV (Capability Maturity Model Integration for Development) and MR-MPS (Reference Model for Software Process Improvement). This analysis of the above-used standards provides best practices through specific characteristics necessary to model and represent the software process. As we have not used the QMs such as CMMI-DEV

^{††}<https://www.academia.edu/>

^{‡‡}<https://www.scirp.org/journal/index.aspx>

or MR-MPS in the article, the modeling constructs of various LCDPs are not semantically compared in a standard way. However, an LCDP can easily interpret any core elements of BPMN.

In Reference 15, various standard process structures with modeling elements are considered. These modeling elements are Process, LifecycleModel, Combination, Activity, Artifact, Resource, Procedure, PatternofActivity, and Restriction. With these software structural (modeling) elements, SPEM and BPMN notations correspond to the two QMs (CMMI-DEV and MR-MPS). Further representativeness of SPEM and BPMN notations are analyzed by considering characteristics such as expressiveness, reuse, management, evolution, multilevel, understanding, and organizational integration. Finally, various modeling elements of BPMN are discussed and matched with the low-code platform. These matching are based on quality criteria such as expressiveness, understandability, conformity to standards, granularity, executability, and orchestability.¹⁴

In Reference 16, the authors select a business process modeling language by analyzing the multi-criteria depending on the modeler's requirements. These criteria are expressiveness, flexibility, formality, readability, support tools, usability, and ease of learning. The mentioned criteria help to evaluate different business process modeling languages such as BPMN, Event-driven Process Chain (EPC), Petri Net, UML-Activity Diagram (UML-AD) and Yet Another Workflow Language (YAWL). The modeler can provide different weights to the criteria based on the modeler's requirements which will be calculated to select the most appropriate modeling language. Similarly, paper¹⁷ also uses an extensive multi-criteria evaluation method to choose a sensitive business process modeling to improve the localization, identification, and characterization of the most critical knowledge. Our current work does not provide any weights to the quality criteria of the modeling construct of BPMN and that of the various LCDPs. However, the modeler can give the weights according to her requirements in developing a particular application using a specific LCDP. For example, a developer may give more importance to the expressiveness of the modeling construct and less to the readability or the ease of learning criteria. On the other hand, a naive non-developer can give more importance to the readability and ease of learning and less to the expressiveness of a modeling construct.

Wang et al.¹⁸ discusses some major business process modeling standards by considering quality criteria such as meta-model, graphical notation, serial representation, extensibility, and tool support. The considered business process modeling approaches in this article are BPEL4WS (Business Process Execution Language for Web Services),¹⁹ BPMN (Business Process Modeling Notation), UML (Unified Modeling Language),²⁰ XPDL (XML Process Definition Language),²¹ Petri Net²² and IDEFO (Integration Definition Method) and IDEF3.²³

In Reference 24, the strength and weaknesses of the several business process modeling languages are discussed so that a comparative perspective can be drawn among them. The comparative framework is based on relevant criteria such as expressiveness, readability, usability, user friendly, formality, versatility, universality, tools support, and flexibility. These criteria are compared among the five process modeling languages such as BPMN, Event-driven Process Chain (EPC), UML-Activity Diagram (UML-AD), Role Activity Diagram (RAD), and Integration Definition (IDEF). The discussed framework quantifies the level of support that each process modeling language is chosen based on the values of each criterion.

In Reference 25, the selection of a process modeling is characterized by several quality criteria, such as intelligibility, the ability to express process elements and workflow patterns along with the software support, and portability. Furthermore, the mentioned quality criteria characterize the modeling languages such as BPMN, EPC, and UML. A comparative analysis is done to unify models and standards by creating a unified process that could be useful in other projects in the industry.

In Reference 26, authors evaluate the usability of BPMN and UML-Activity Diagram (UML-AD) for many business users based on an empirical comparison to justify their ability to express and communicate business process semantics. Furthermore, the article claims that process modeling through UML-AD is similar to that of BPMN in terms of user effectiveness, efficiency, and user satisfaction in a specific context of architecture development scenarios.

3.3 | Key aspects that are considered when comparing business process modeling languages

By referring to Table 2, this section addresses the research question underpinning the performed study. In particular, we analyze the related works that have been summarized in the previous section with respect to different qualitative characteristics. According to the performed analysis, the broader criteria to differentiate process modeling languages are *conciseness*, *executability*, *standardization of process models* such as UML or BPMN, *granularity* and *collaborativeness*.

TABLE 2 Quality criteria to compare process modeling approaches in the literature

Quality criteria	Existing studies
Conciseness	12,14,16-18,24-26
Executability	10,12,14,18
Standardization of process models	12,14,15,18
Granularity	12,14-16,18
Collaborativeness	14,15

3.3.1 | Conciseness

Two kinds of properties focus on conciseness. They are *semantic* and *syntactic* properties. Semantic properties enable the richness in explaining the different states of process models that can be replicated in advanced complex real scenarios and reducing the redundant features to showcase process models. In contrast, the syntactic properties focus on the user-friendliness of the process modeling tool. Papers such as Bendraou et al.,¹² Garcia et al.,¹⁴ Portela et al.,¹⁵ Guizani et al.,¹⁶ Hassen et al.,¹⁷ Pereira et al.,²⁴ Kelemen et al.,²⁵ and Birkmeier et al.²⁶ show key importance in the expressiveness and the understandability of the process modeling language. They also studied the modularization, reusability, and universality of the modeling language.

3.3.2 | Executability

The execution of the business process models is integral to developing a process-oriented application. For example, an LCDP focuses on executing the process models using data models and logic applied in creating an application. Also, the execution of BPMN specifications is done by transforming models to an XML-based business process execution language (BPEL) that helps to optimize and automate the defined business processes.¹⁰ But the use of BPEL is only sometimes used nowadays. Executability criteria are considered in papers such as Bendraou et al.¹² and Garcia et al.¹⁴

3.3.3 | Standardization of process models

The standardization and formalization of process models showcase the clarity in defining particular process model elements, which can be used for large groups of people. There are papers such as Bendraou et al.,¹² Garcia et al.,¹⁴ and Portela et al.¹⁵ that consider standardization as one of the criteria to evaluate different process models.

3.3.4 | Granularity

A process model can be subdivided into logical and syntactical forms based on user demands. This granularity can be either coarse-grained or fine-grained based on the complexity and the user's requirement to build a particular process model. For example, papers such as Bendraou et al.,¹² Garcia et al.,¹⁴ Portela et al.,¹⁵ and Guizani et al.¹⁶ use granularity to characterize various process modeling language.

3.3.5 | Collaborativeness

Multiple people can work to build on the same process model while maintaining the cohesiveness of the work done, especially from multiple remote locations. For example, papers such as Garcia et al.,¹⁴ and Portela et al.¹⁵ allow collaboration to validate and build process models in real environments.

To summarize on comparing modeling constructs of different process modeling tools, we have used different quality standards defined in the paper.¹⁴ Gartner forecasts that market for LCDP would grow by 23% in 2022–2023.²⁷ To assist this growth, it is important to understand the data-handling and process modeling constructs used in different LCDPs.

Therefore, the study to compare LCDP and BPMN focuses on using data-handling capabilities in their process modeling framework that are executable to perform various application tasks.

The performed analysis of the related work shows various methods in which several defined quality standards are formulated to define a framework that can compare multiple modeling constructs. No quantitative analysis has been done to compare different process modeling tools. It is worth noting that previous works have yet to consider modeling constructs of any LCDP and compared them with other modeling technologies.

4 | LOW-CODE DEVELOPMENT PLATFORMS

LCDPs are defined as “platforms that enable rapid delivery of business applications with a minimum of hand-coding”.¹ According to a Gartner report,⁵ leading low-code platforms are OutSystems, Mendix, Microsoft Powerapps, and Salesforce Lightning. In this section, we consider all of them in addition to the recent and niche⁵ platforms such as Amazon Honeycode, Google Appsheet, Zoho Creator, and Thinkwise. Most of these platforms use similar concepts in graphical user interfaces, allowing developers to define and manipulate data specified through tables, forms, reports, and other kinds of representation. LCDPs permit importing data tables from external databases or creating new ones using the platform’s data models. Data models or imported data tables can be edited, which then makes a form screen where the application user gives input to it. The input data is then shown in a report screen, which can still be edited/updated by the user. Lastly, the input can be manipulated to give output in charts, graphs and so forth. The form and the report can be edited or managed using UI controls such as link icons, buttons and so forth. These generalized basic steps in using LCDPs to develop applications can be further extended by using different platform-dependent facilities to manage interoperability and collaboration.³

This section discusses different LCDPs by referring to the previously presented “Recruitment Drive” example. For the sake of presentation and explainability, we focus on modeling explanatory parts of the running example. In particular, we focus on the following steps: on clicking the user’s application link, the detail and the user’s eligibility information are stored in the database. If the eligibility information given by the user equates with the considered eligibility criteria of a post set by the company, then the recruitment system application would send a pass message to the user; otherwise, the user will get a fail message.

The considered scenario is assumed to rely on two data tables, namely Applicant and Recruitment Notification. Therefore, the implementation of the case study with the LCDP at hand starts with the creation of two tables. An Applicant table consisting of the fields *Name of the applicant*, *Email Id*, *Eligibility Criteria* and *Post Applied*. A Recruitment Notification table has to be also created with the fields *Name of the Post* and *Eligibility Criteria*. The table Applicant is linked to the Recruitment Notification through the foreign key *Post Applied*. This also means that the *Post Applied* of the Applicant table is a lookup variable for the *Name of the post* field inside the Recruitment Notification table. Each applicant can apply for one or more posts in the Recruitment Notification table. A user on a post fills an application form which is then stored in the Applicant table. Suppose the *Eligibility information* of the applicant matches with the *Eligibility criteria* mentioned in a specific *Post name* row in the Recruitment Notification table. In that case, a notification email/message is sent to HR/applicant mentioning the applicant success report. If the *Eligibility information* of the applicant does not match the *Eligibility criteria* then the whole row of that particular applicant is deleted from the Applicant table. Other minor details of the case study workflow are not considered as it would be repetitive to model similar business process elements with similar logic.

As anticipated, in the following sections, we discuss the specification of the described case study given with different LCDPs. It is worth noting that we are comparing BPMN modeling elements to build the case study application (shown in Figure 1) with its equivalent modeling constructs using different LCDPs associated with the data handling mechanism and their implementation. The analysis is presented from three different points of views, that is, the *core* modeling elements of BPMN which is then compared to the implementable *workflow* of the eight considered LCDPs, in addition to their specific *data* handling mechanisms.

4.1 | OutSystems

OutSystems²⁸ is a low-code development platform offering a visual, model-driven development environment with industry-leading AI-based assistance. The development environment provided by OutSystems is a suite consisting of Service studio and Integration Studio.

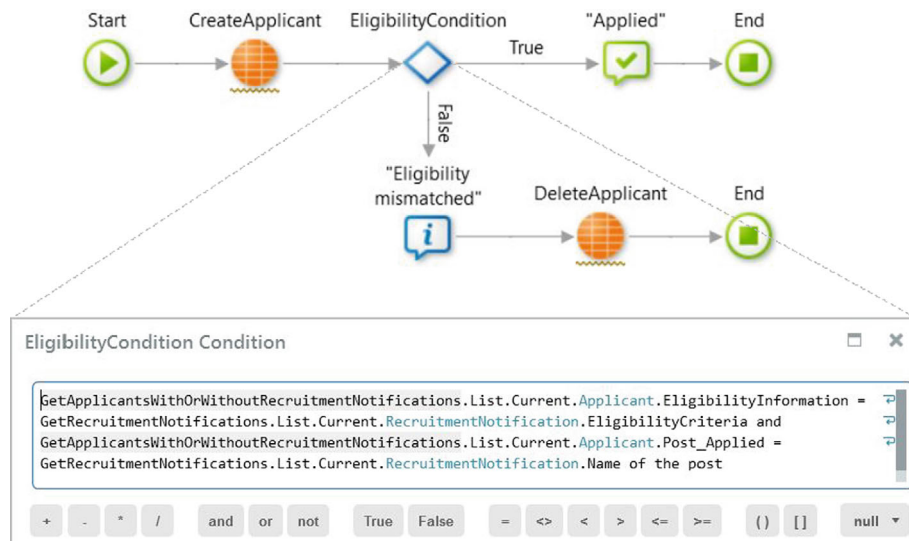


FIGURE 2 OutSystems business flow

4.1.1 | Core modeling elements

OutSystems supports BPMN Event by means of different modeling elements including *Start*, *Conditional Start*, *End* and *Wait*. Activity is referred to as *Human Activity*, *Automatic Activity*, and *Execute Process*. Also, Gateway, Message and Annotation are supported by means of the modeling constructs *Decision*, *Send Email*, and *Comment*, respectively.

4.1.2 | Workflow modeling

Workflows defined in OutSystems typically handle different business processes such as invoices, processing orders, or handling complaints, also known as BPT (Business Process Technology).²⁸ The key feature in developing business processes in OutSystems is the separation of the process development built by *steps*, *interfaces*, *logical structure*, and *databases*. Steps define the flow of the process under development. Interfaces enable different ways to access the application. For instance, interfaces enable the parts of the app to be shown in the UI, for example, charts, forms and so forth. Logical structures build reusable logic blocks and reusable integration components like SOAP or REST services. Databases are used to manage and store data, even from external sources.

4.1.3 | Data handling capabilities

In the data section of OutSystems' user interface Service Studio, a table can be created by adding an entity to the in-built cloud database or importing a table from Excel. When the table is imported from Excel, it can be transformed into the user-defined editable table in the platform if needed. Moreover, querying and fetching individual data from an Excel sheet is not possible directly from/to the external data without using an external API. A developer manipulates data or attributes of the table only after importing the table to the OutSystems environment. Integration of other external databases is done in *Integration Studio*^{§§} by configuring the extension to use a database connection and then refer and use it. Such integration is an interoperability feature of the platform and would require API integration with some coding; therefore, these integrations can be challenging for inexperienced developers.

Figure 2 shows the OutSystems user interface Service Studio at work, which is the development environment that users can use to model business applications.²⁹ Figure 2 shows the specification of the applicant eligibility explained above with a dedicated scripting language. The reported script assures that if the stated eligibility condition is matched,

§§ https://success.outsystems.com/Documentation/11/Reference/Integration_Studio

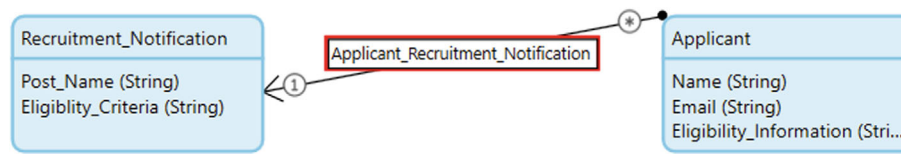


FIGURE 3 Recruitment drive domain model in Mendix

the applicant and the HR manager receive a message that the applicant successfully applied for the post. On the other hand, if the eligibility condition does not match, the current row in the “Applicant” table is deleted, meaning the applicant cannot apply for the chosen post.

4.2 | Mendix

Mendix^{¶¶} offers both visual-modeling and low code integration functionalities. It uses a dedicated modeling tool called Microflows to express the application’s logic.

4.2.1 | Core modeling elements

Mendix has their Event element termed as *Events* only and Activity is termed as *Activities* and *Error handlers*. Also, Gateway, Data Objects, Sequence Flow and Association are termed as *Splits*, *Input parameters*, *Flows*, and *Connectors*, respectively.

4.2.2 | Workflow modeling

The process of creating a new application in Mendix starts with the creation of the data model. Then, the user needs to edit the fields of the entity/table of the app. This is followed by creating an object linked to an entity that needs to be managed to add/update records. Once these steps are completed, the user can build processes employing the available Microflow modeling language. Microflow is based on the BPMN standard, and it helps to extend or change the default behavior of the developed application.³⁰ Microflow also defines several business processes and integrates with external systems, databases, or web services. Microflows decide the flow of the defined pages involved in creating the logic for the application. This would extend or create new business processes and show the application’s extensibility by integrating external services and databases.

4.2.3 | Data handling capabilities

The table is created by designing a domain model inside the Mendix platform. In this domain model, the user can define all the attributes and relationships among the specified table. Although it is possible to connect it to an external database, it requires separate import modules to connect to different data sources. However, it is not easy to use such import modules for non-expert developers.

Figures 3 and 4 show fragments of the data model and of the workflow specification given in Mendix, respectively, of the explanatory example. According to the data entities defined in Figure 3, an applicant can apply for exactly one post at a time, as can be noticed from the target cardinality of the association from the entity Applicant to Recruitment_Notification. The defined data entities and their relationships underpin the specification of corresponding form pages enabling the execution of CRUD (create, read, update, delete) operations. For instance, the form related to the Applicant entity refers to the values of the attribute *Post_Name* that can be retrieved by a drop-down variable attained from the Recruitment_Notification table.

¶¶ <https://www.mendix.com/>

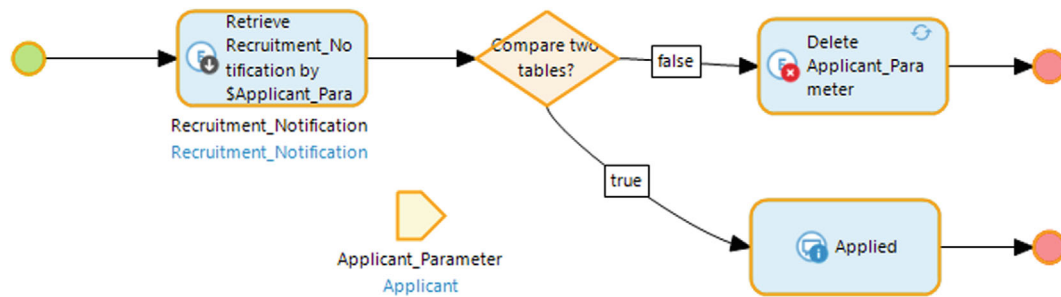


FIGURE 4 Recruitment drive microflow in Mendix

Microflow allows specifying workflows as shown in Figure 4. In the example, the entity *Applicant* of a particular row is parameterized, and the associated data linked to a specific row in the *Recruitment_Notification* entity is retrieved. Then, the decision flow is used to compare the entered eligibility information in the *Applicant* form with the eligibility criteria of the *Recruitment_Notification* form. If they match, a message is shown to inform the applicant that the application has been successfully applied. Otherwise, the current row in the *Applicant* list is deleted.

4.3 | Zoho creator

Zoho creator^{##} is a cloud based software to create applications without coding experience or IT expertise. In Zoho Creator terminology, a data table is called *form*, whereas the view of a table is called *report*. The workflow in Zoho is created using a dedicated scripting language called Deluge.³¹

4.3.1 | Core modeling elements

In Zoho Creator, there are multiple elements matching with BPMN Event. In particular, the first Event setting, for example, “On a form event,” “In a schedule data,” “On an approval activity,” “On a payment activity,” and “On a function call,” can be set by associating the action with the selected event. The second Event setting is called “Record Event,” that are sub-divided as “Create,” “Edited,” “Created or Edited,” and “Deleted”. The third Event setting is “Form Event” which is further divided as “Before form submission,” “On form submission,” and “After form submission,” referring to the form submission events. Likewise, Activity are broadly sub-divided into seven parts. They are “Data Access,” “Notification,” “Field Actions,” “Integrations,” “Client Function code,” “Custom Action,” and other “Deluge scripts”^{|||}. Gateway is implemented in terms of conditional statements provided by Deluge scripting language. Finally, “Data Store” is cloud-based and can be edited using data access code available in Deluge script that perform actions such as add record, fetch records, aggregate records, update records, for-each record and delete records.

4.3.2 | Workflow modeling

In Zoho Creator, rules and workflows can be defined by writing custom Deluge code to determine the behavior of an element of an application’s page. The behavior construct can be conditional (if, else if, else, etc.), data access (add, fetch, aggregate, etc.), or client functions (hide/show, enable/disable, etc.) types. Workflow actions can be triggered by adding, updating, validating, or deleting page elements. Loading of forms can also be a possible workflow trigger. Also, button clicks and user inputs can trigger the execution of scripts building up the application. In Zoho Creator, there are three types of workflows: (i) stand-alone scripts are used as a function to perform a task; (ii) time-based scripts are used to schedule a task; (iii) component-based scripts combine the first two workflow types that perform necessary actions on the forms/tables of interest.

^{##}<https://www.zoho.com/>

^{|||}<https://www.zoho.com/deluge/>

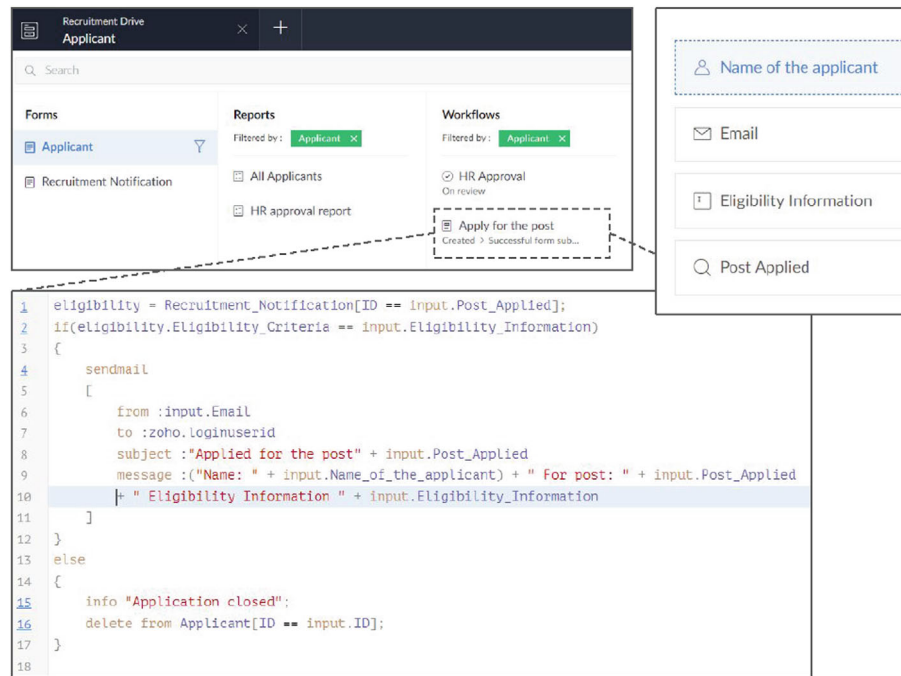


FIGURE 5 Recruitment drive in Zoho Creator

4.3.3 | Data handling capabilities

A table can be created by developing the form in the Zoho Creator environment, which is used to get the user input and automatically generate a list/report page. Alternatively, a developer can import an Excel or CSV file transformed into the platform's environment according to the developer's need to manipulate the required fields or data if needed. No direct manipulation of the external file is possible in such importing of data.

Based on the implementation description of the case study recruitment drive, two forms (i.e., tables) are created, namely *Applicant* (shown for convenience on the right-hand side of Figure 5) and *Recruitment Notification*.

As specified in the script reported at the bottom of Figure 5 the value of *Post_Applied* is retrieved from the *Recruitment Notification* form. The eligibility information in the *Applicant* table is equated with the eligibility criteria field in the *Recruitment Notification* form. If they match, then an email is sent to HR mentioning the detail for the *Applicant* form. If they do not match, then the whole current input row of the *Applicant* table is deleted.

4.4 | Microsoft PowerApps

Also Microsoft has its own low code platform, and it is called PowerApps^{***}. In Microsoft PowerApps, the workflow can be managed by using an OCL-like (Object Constraint Language) language³² that can be applied to different objects of the modeled screens/pages as described below.

4.4.1 | Core modeling elements

In Microsoft PowerApps, the BPMN Event element has the equivalent *Phase* construct, whereas *Activity* instances are developed with OCL (to manipulate or perform an action inside a given page), or in terms of *Workflow*, *Action Step* and *Flow step* elements to perform sequential activities. Also, the BPMN Gateway element is equivalently executed by the conditional element *if* and *switch* statement written in OCL code. Also, multiple connected data objects are used to

***<https://powerapps.microsoft.com/>

connect different Data Stores such as MS SQL, Sharepoint, OneNote, OneDrive and so forth that correspond to the Data Store element in BPMN.

4.4.2 | Workflow modeling

In Microsoft PowerApps, there are three ways to model a workflow when building an application: (i) the user can use multiple screens to develop complex forms consisting of different parts and pages having different formulae and controls; (ii) the user can use a single page with complex OCL (Object Constraint Language) code; (iii) a hybrid approach employs a single page, which integrates the business logic from external services such as MS Power Automate, Zapier and so forth. In particular, MS Power Automate is the tool integrated into MS PowerApps for profiling business process steps and link them in the formulae of the defined fields or buttons according to the modeled logic.³³ MS Power Automate performs five types of workflows that can be incorporated into the PowerApps platform. They are *automated flow*, *instant flow*, *scheduled flow*, *user-interface flow*, and *business process flow*. Automated flows get executed once the corresponding trigger occurs, whereas instant flows will be executed based on the defined user's actions. Scheduled and user-interface flows work at a time interval or by clicking dedicated buttons on the application user interface. Finally, business process flows work on the parallel or sequential event-based actions defined by the developer.

4.4.3 | Data handling capabilities

The connection to external databases (such as Microsoft Sharepoint, Dynamic 365, OneDrive, Salesforce, etc.) is one of the possible options offered by the PowerApps platform. Such a connection creates a common data model. Therefore, the developer can change the table's metadata (attributes, relationships, and data types). The user can fetch and manipulate data directly to and from the common data source. Those external data sources are linked with the PowerApps platform that requires authentication in all the associated services or databases. Custom-built tables can be exported to external databases if needed.

In the example shown at the bottom of Figure 6, the OCL-style language is used to specify the logical workflow of comparing the required eligibility with the application data, which are provided through a designed form. In case of the eligibility condition between the Applicant and the Recruitment Notification table matches, the HR will be notified; otherwise, the current user input gets deleted. Then, the tables can be linked using the data connector used in the platform that connects different data stores such as Microsoft SQL, Sharepoint, OneDrive and so forth.

4.5 | Salesforce lightning

The Lightning platform^{†††} is the low-code answer to app development by Salesforce, a leader company in CRM technology. In Salesforce Lightning, workflows can be defined in four different ways, that is, by using *Flows*, *Workflow field update*, *Process builder*, and with the help of *Apex code*.³⁴ In this article, we focus on using *Flows*. *Workflow field update* and *Process builder* are limited in their ability to write workflow rules to create or delete records. In contrast, workflows through *Apex code* are supported only in the professional version of Salesforce Lightning, and it requires complex coding; thus, not favorable for inexperienced users.

4.5.1 | Core modeling elements

Salesforce Lightning platform has their BPMN Event element termed as *Start*, and *Stop*. The BPMN Activity element is instead supported by platform specific modeling elements such as *Add Object*, *Immediate Actions*, and *Scheduled Actions*. Finally, Gateway is referred as *Add Criteria* in the Lightning platform.

^{†††}<https://www.salesforce.com>

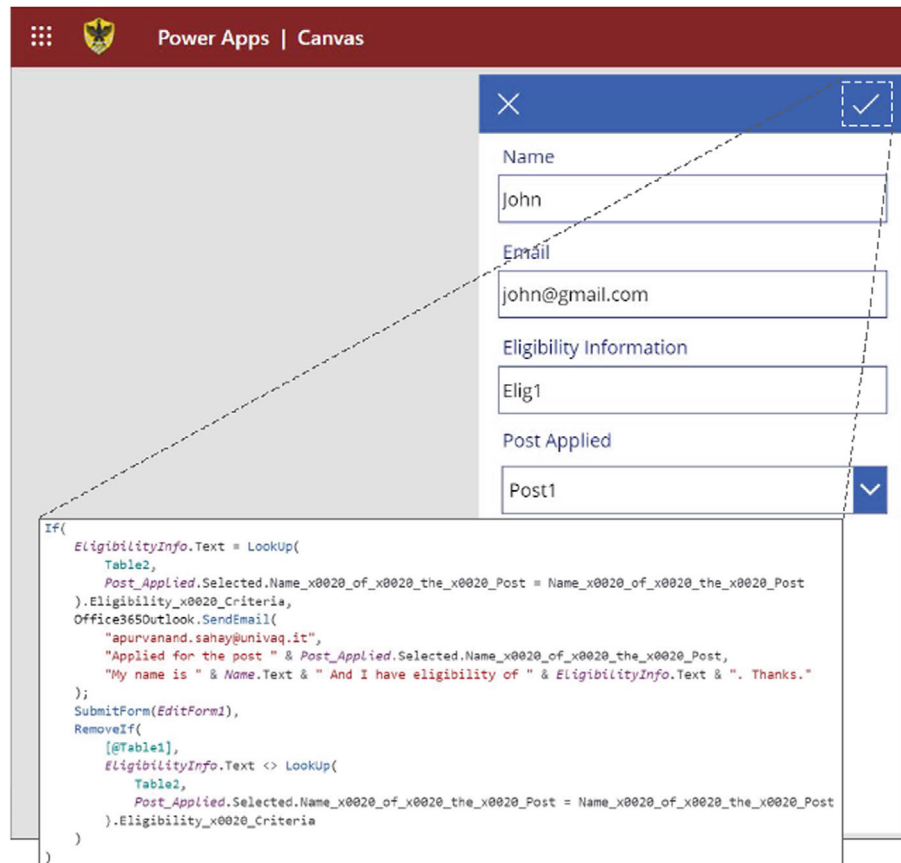


FIGURE 6 Recruitment drive in Microsoft PowerApps

4.5.2 | Workflow modeling

New processes can be created when the user accesses the “Process Automation” function in the Salesforce Lightning platform. In addition, the business workflow template offers a way to link the action to an event. For example, an event can be “a record changes” for building a particular type of step when a record (row) changes in the object (table). Or it can be processed by calling another process (also called the nested process). Also, the flows mechanism is available in Salesforce Lightning to develop business logic through a drag-and-drop interface that provides the user with business modeling elements that include events, activities, and actions.³⁵ Finally, Apex coding can be employed to update fields during workflow executions. However, such languages are not used extensively by non-developers, and their adoption has pros and cons, as discussed in Reference 34.

4.5.3 | Data handling capabilities

In this platform, a table or an object can be created by using the Object Manager^{***} that could be used to define the schema of various tables, which underpin the data of the application under development. Alternatively, a table can be imported from an external spreadsheet that allows manipulation only inside the Salesforce environment. Authentication is required to connect external spreadsheets/databases.

The workflow shown in Figure 7 depicts the *Flows* that get executed when a record is created in the Applicant table. The condition is set to equate the table Applicant’s eligibility information with the post applied to the row with the same post name in the table Recruitment Notification’s field of eligibility criteria. The eligibility

***https://help.salesforce.com/s/articleView?id=sf.extend_click_find_objectmgmt_lex.htm&type=5

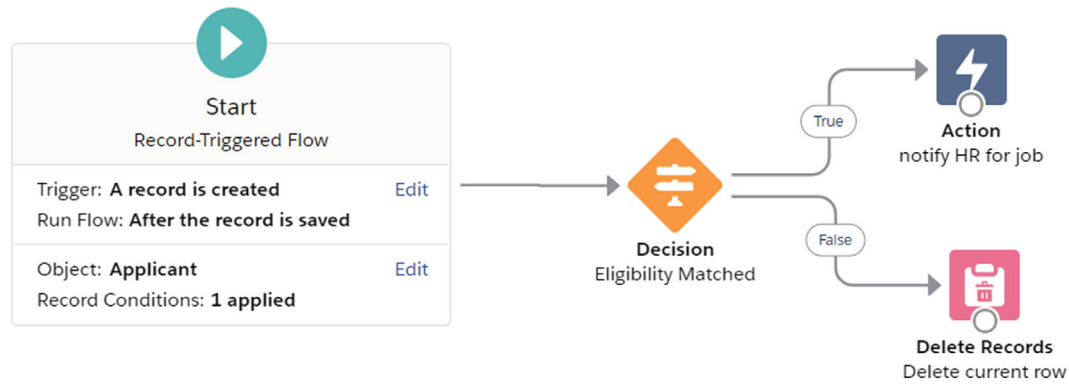


FIGURE 7 Recruitment drive workflow in Salesforce Lightning

condition is specified in the decision activity as follows. `!$Record.Eligibility_Information__c Equals !$Record.Post_Applied__r.Eligibility_Criteria__c`. If the condition is true, then the HR and applicant will be notified about the applicant's success; otherwise, the current applicant's row is deleted.

4.6 | Thinkwise

Thinkwise^{§§§} is a LCDP offering modeling tools available on desktop, web, and mobile devices typically useful for large scale business software^{¶¶¶}.

4.6.1 | Core modeling elements

Thinkwise provides modelers with a BPMN-like notation to specify workflows as compared to the other LCDPs. The BPMN element Lane is defined by creating *user roles*, whereas Events, Activities and Gateways are defined by the *activity* elements that modelers can specify in a dedicated environment for specifying process flows. Also Sequence Flow, Message Flow and Association are supported to connect specified activities. All the above-mentioned modeling elements get executed by the Thinkwise execution environment. Several programming languages are supported to define business logic including SQL, R, Python, and Java.

4.6.2 | Workflow modeling

In the Thinkwise platform, process flows and business logic specifications are distinguished. A process action handles the process flow to create events, triggers, and decisions and their interactions via process steps.³⁶ In contrast, business logic is developed using the supported programming languages to handle data and execute controlled procedures. Furthermore, user roles can be defined to protect the application and enable its usage according to the user's role, such as developers, business analysts, users and so forth. Along with intuitive graphical workflows, code-based business logic is also possible in Thinkwise, which supports different languages, including SQL, R, and Java.

4.6.3 | Data handling capabilities

In Thinkwise, tables can be created by using the data model provided in the Software Factory^{###} where the fields and data can be created or modified. Once the table of interest is created, data manipulations can be implemented through

§§§ <https://www.thinkwisesoftware.com/>

¶¶¶ <https://www.thinkwisesoftware.com/en/overview/>

https://docs.thinkwisesoftware.com/docs/sf/sf_general.html

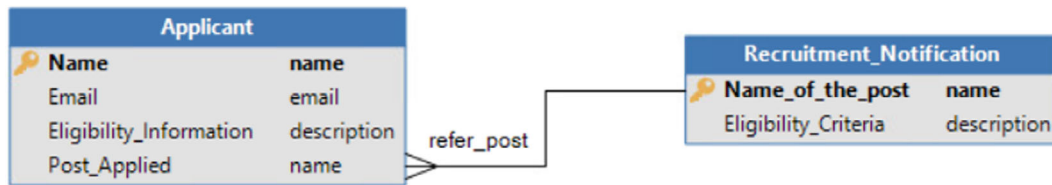


FIGURE 8 Recruitment drive data model in Thinkwise

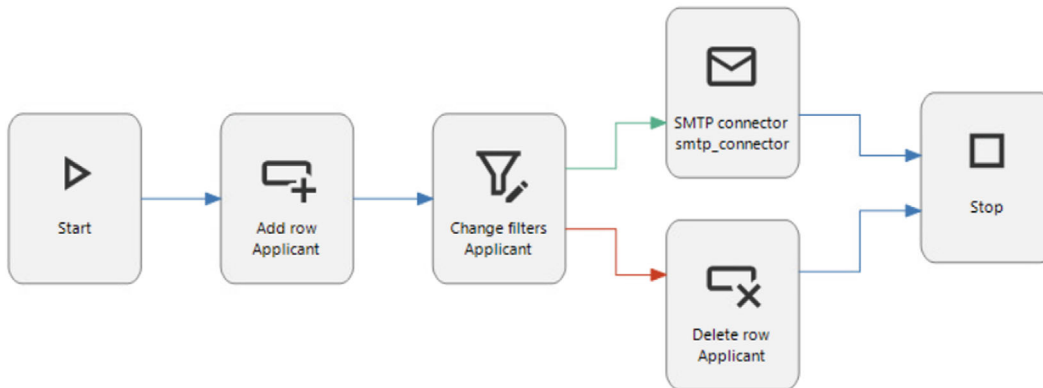


FIGURE 9 Recruitment drive process flow in Thinkwise

external databases such as DB2, SQL Server Management Studio, and Oracle. This means that the data handling occurs by interacting with the instance of the used external databases.

The Thinkwise data model related to the running example is shown in Figure 8, whereas the related process flow is depicted in Figure 9.

The SQL code implementing the business logic of the considered case study is shown in Listing 1.

```
SELECT a.Name, a.Email, a.Eligibility_Information, a.Post_Applied
FROM Applicant a, Recruitment_Notification r
WHERE a.Post_Applied=r.Name_of_the_post AND
a.Eligibility_Information=r.Eligibility_Criteria
```

Listing 1: Logic for the case study recruitment drive

Such a mixture of code-based and graphical-based business logic allows the business people and IT teams to collaborate and work in an agile environment to build an application. For example, the Figure 9 depicts the business workflow that compares the eligibility information from the Applicant table to the eligibility criteria from the Recruitment_Notification table. If this condition is true, a success message is sent to the applicant/HR; otherwise, the applied person's record is deleted from the table Applicant.

4.7 | Google Appsheet

Appsheet^{|||||} is the LCDP from Google to build apps and support work automation.

|||||<https://www.appsheet.com/>

4.7.1 | Core modeling elements

In Google Appsheet, the BPMN Event element is mapped to events that are categorized as “New Workflow Rule,” “When this happens,” and “If this is true and Do this.” Activity is broadly termed as *Action Reaction* that acts on different types of objects, including Email, Notification, SMS, TakeAction, Webhook, and MakeDoc. Lastly, Gateway can be mapped to the events and conditions declared under the events “When this happens,” and “If this is true and Do this.”

4.7.2 | Workflow modeling

Workflow automation in Google Appsheet comprises three types of actions.³⁷ The first type is based on a data change that triggers an event when data is captured through the built application. The second type is scheduled on a specific date/time, and the third type is based on webhooks that can be scheduled or activated on data’s additions/updates/deletions. Google Appsheet supports the connection to external services, including Zapier and IFTTT. Pre-built workflows are also available in Appsheet that can be reused to specify a newer workflow for a different app.

4.7.3 | Data handling capabilities

Tables can be imported from external data sources from different vendors such as Google, Microsoft, Dropbox, Smartsheet, and Airtable. The data source is directly linked to the platform. This means that the data is defined (attributes, relationships, references, keys) and manipulated (search, filter, add, delete, etc.) by the spreadsheet formulae or by defined process workflows on Appsheet that sync with the used database.

In Appsheet, both tables Applicant and Recruitment Notification were defined and imported from Google Sheet, that is, the spreadsheet tool provided by Google. The workflow is implemented by selecting the *Behavior* menu in the Appsheet Web development tool (see left of Figure 10). Then, the current applicant’s data in the Applicant table is chosen by selecting the “Target data” field in the form. On such a table, the developer can add and update events such as mentioning the business logic [Eligibility Information] = [Post Applied] . [Eligibility criteria] as specified in the field condition of Figure 10. If this condition is true, then the necessary task(s) is performed. In our case, the task is to send the email to HR/applicant. Alternatively, if the condition [Eligibility Information] <> [Post Applied] . [Eligibility criteria] is applied (Here, “<>” means *is not equal to*), then the delete action is performed on the whole row managed by AppSheet that leads to the deletion of the row back to the Google Sheet.

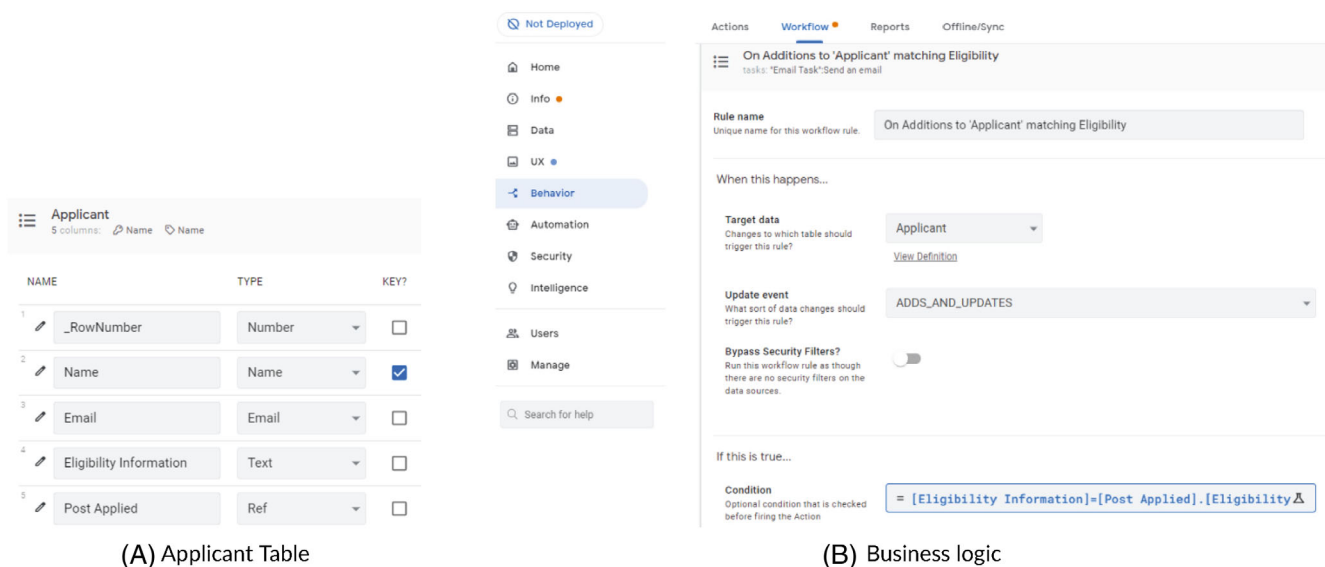


FIGURE 10 Recruitment drive process flow in Google Appsheet. (A) Applicant table; (B) business logic

4.8 | Amazon Honeycode

Honeycode^{****} is the LCDP released by Amazon in June 2020 to support the development of Web and mobile applications.

4.8.1 | Core modeling elements

The Amazon Honeycode platform has their Event termed as *Start when clicked*. An Activity can be mapped to spreadsheet formulae to perform an action, for example, *Add row to* and *Take data from and write to*. Finally, the Gateway can be implemented using platform specific SQL-like statements that include *filter* and *if* conditions.

4.8.2 | Workflow modeling

Amazon Honeycode uses spreadsheet data to create an application. An application is created by defining a spreadsheet which can then be transformed into list or report elements. Such elements show all the items in the built application along with a detailed view of an individual data element of the application and allow the user to use the form screen to give input to the application.³⁸ Further, the application's business logic is managed by spreadsheet formulae, which can add necessary actions such as notify, add a row, delete a row, overwrite an element in the screen, update or insert a row, and navigate another screen.

4.8.3 | Data handling capabilities

Honeycode manages tables by importing CSV files or creating a blank table directly following the platform's wizard. Any data manipulation (add row, filter, search, etc.) can be done using Excel-like formulae within the platform.

In Amazon Honeycode, the tables Applicant and Recruitment Notification are created, followed by their respective screens (forms and list). At the submit button of the form, the condition specified (right-hand side of Figure 11) is used to validate the applicant's job application by matching the input Eligibility_Information with the Eligibility_Criteria for that post mentioned in the Recruitment Notification table. If this matching is true, then the HR/applicant is notified. Alternatively, if we use inequality (<>) in applicant's eligibility information with recruitment notification's eligibility criteria, we can delete the entire row. This will allow only those applicants to be considered for the chosen post with the required eligibility criteria as mentioned in the script at the bottom right side of Figure 11.

5 | DISCUSSING LCDPS WITH RESPECT TO BPMN MODELING CONSTRUCTS

Table 3 shows the coverage of the BPMN constructs overviewed in Table 1 from the previously analyzed LCDPs.^{†††} According to the performed analysis, depending on the adopted LCDP, the specification of workflow processes can or cannot follow a BPMN-style. For example, low-code platforms such as OutSystems, and Mendix provide users with a platform-specific BPMN-style modeling framework. In contrast, Microsoft PowerApps, Zoho Creator, Google Appsheet, and Amazon Honeycode support a non-BPMN-style modeling framework that uses proprietary coding or a hierarchical workflow mechanism to define business flows. Other platforms such as Salesforce Lightning and Thinkwise support both BPMN and non-BPMN styles.

The combination of process and data defined and managed with an intuitive visual interface with minimal coding describes the peculiar characteristics of any LCDP, which is supposed to support the fast development and management of business applications. In Section 4, the considered LCDPs are further discussed by considering for each of them the BPMN core elements covered and the mechanisms provided to model workflows and handle corresponding data. Based

**** <https://www.honeycode.aws>

†††† The complete analysis is available at https://docs.google.com/spreadsheets/d/1lllnNTwXNgQCXPVbaRBAk_dIs6gPVd0CNxub_MBQw/edit?usp=sharing

FIGURE 11 Recruitment drive process in Amazon Honeycode

TABLE 3 BPMN workflow constructs covered by the analyzed low-code platforms

BPMN elements	OutSystems	Mendix	Zoho Creator	PowerApps	Lightning	Thinkwise	Appsheets	Honeycode
<i>Swimlanes</i>								
Pool								
Lane						✓		
<i>Flow object</i>								
Event	✓	✓	✓	✓	✓	✓	✓	✓
Activity	✓	✓	✓	✓	✓	✓	✓	✓
Gateway	✓	✓	✓	✓	✓	✓	✓	✓
<i>Data</i>								
Data object		✓	✓	✓			✓	
Data store			✓	✓	✓	✓	✓	✓
Message	✓	✓						
<i>Artifacts</i>								
Group								
Annotation	✓	✓						
<i>Connecting objects</i>								
Sequence flow		✓			✓	✓		
Message flow					✓	✓		
Association		✓			✓	✓		

TABLE 4 Main mechanisms to specify workflows in low-code development platforms

	Code	Hierarchical structure	Automated BPMN	User-defined BPMN	Hybrid
OutSystems				✓	
Mendix				✓	
Zoho Creator	✓				
Microsoft PowerApp	✓	✓			✓
Salesforce Lightning	✓		✓	✓	✓
Thinkwise	✓			✓	✓
Google Appsheet		✓			
Amazon Honeycode	✓				

on the discussed core modeling elements across several LCDPs and the content of Table 3, it is noticeable that although there may not be one-to-one similarities between the BPMN and the LCDPs modeling elements, an LCPD is capable of expressing workflows (similar to BPMN) and the associated business logics in different ways while developing business applications.

To make an overview of the different types of business workflows that might be specified with the analyzed platforms, we can distinguish three ways of defining business processes: (i) use of drag-and-drop BPMN-like tools to create business flows and logic, (ii) non-BPMN-like tools that use either some programming language or some hierarchical decision-making structure, (iii) combination of the first two means known as a hybrid category. Thus, as shown in Table 4, we can distinguish five main mechanisms to specify workflows in LCDPs as described in the following:

1. *Code-defined modeling*: A business process is created using a proprietary or a known programming language such as Java, Javascript, SQL and so forth. LCDPs such as Zoho Creator and Salesforce Lightning platform support proprietary Deluge code and Apex, respectively. Also, Thinkwise supports multiple languages such as SQL, R, Java and so forth to define business logic.
2. *Hierarchical structure of business process*: A business process is created with a tree-shaped business flow, causing one event to follow another based on some conditions or trigger points. LCDPs such as Google Appsheet or Microsoft Power Automate in PowerApps support a hierarchical-based structure for business flows.
3. *Automated BPMN-like structure*: A business process is created by an automatically defined BPMN structure where the user can fill the events, objects, or conditions based on the pre-defined BPMN flow. Salesforce Lightning platform supports automated and user-friendly specifications of complex business modeling.
4. *User-defined BPMN modeling*: A business flow is created by a BPMN-like experience defined by the user based on her requirements. LCDPs such as OutSystems, Mendix, Thinkwise and so forth support BPMN-like modeling that expresses an extensive range of business flows.
5. *Hybrid modeling*: Business flows are partially implemented using two or more listed business process types. LCDPs such as Salesforce Lightning platform, Microsoft PowerApps, and Thinkwise support multiple workflow mechanisms for the same working business flows. Salesforce Lightning platform uses automated BPMN and code-based modeling in Apex language. Microsoft PowerApps uses a hierarchical structure in Microsoft Automate and OCL-based coding. Thinkwise uses a user-defined BPMN for the business flow and SQL coding for the business logic.

Concerning the way data are managed by LCDPs, it is possible to categorize them in two ways as shown in Table 5 and explained below.

1. *Data import*: LCDPs can import data structures from external sources such as Microsoft Excel, Azure, SQL and so forth. In this category, all the structural definitions of the data table, such as attributes and relationships, are imported into the considered LCPD, and they are transformed into the host LCDPs structures to create pages such as forms, lists and so forth. Thus only the table's design is imported, not the corresponding instances. No direct manipulation can be done in the initial data source without using the APIs to integrate them. LCDPs such as Zoho Creator, Salesforce Lightning, Outsystems and so forth are in this category.

TABLE 5 Supported data management

	Data import	Data sync
OutSystems	✓	
Mendix	✓	
Zoho Creator	✓	✓
Microsoft PowerApp		✓
Salesforce Lightning	✓	
Thinkwise		✓
Google Appsheet		✓
Amazon Honeycode		✓

2. *Data sync*: LCDPs can retrieve and store data from and to the considered data sources like Microsoft Excel, Azure, SQL and so forth. LCDPs such as Microsoft PowerApps, Google Appsheet, Amazon Honeycode and so forth import the whole instance of the considered database, and data manipulations are synced back with the initial data source.

6 | DISCUSSING LCDPS WITH RESPECT TO BPMN QUALITY CRITERIA

This section discusses the quality assessment of LCDPs by relying on existing approaches for BPMN. In particular, nine quality criteria characterize, according to Reference 14, the quality of a modeling language: expressiveness, understandability, conformity to standards, granularity, executability and orchestability, measurability, business rules, supporting tools, and validation in real environments. In the following, we discuss such criteria to analyze the quality of the eight considered LCDPs:

Semantic richness and expressiveness: BPMN2.0 defines process models based on core elements such as flow objects, data objects, artifacts, and connecting objects. It is noticeable from Table 3 that all the core modeling elements of BPMN can be interpreted in the LCDP-specific modeling constructs. Such modeling constructs allow a developer to build a workflow for an application that could be executed by applying the business logic of that application. For example, the case study *Recruitment Drive* is modeled using BPMN in Section 2 and its automatized part built by various LCDPs were described in Section 4.

Understandability: The understandability aspect of BPMN is described in Reference 39 in which many modeling guidelines are considered, such as validating models, minimizing model size, applying hierarchical structure with sub-processes, and other optimum use of different modeling elements. These guidelines suggest design decisions that allow the modelers to choose the most relevant modeling tool according to their modeling purpose, such as process learning, information system development and so forth. Compared to the understanding of BPMN, all the eight considered LCDPs' workflow mechanisms are categorized and highlighted in Table 4. Low-code platforms such as Zoho Creator, OutSystems, and Salesforce Lightning support a proprietary code-based and user-defined BPMN-like modeling workflow mechanism. The workflow is designed in an easy-to-use built-in modeling mechanism that could either use BPMN-like modeling constructs such as in Salesforce Lightning or use a combination of different hierarchical or BPMN-like or code-based *hybrid* modeling constructs. The combination of varying modeling constructs in LCDPs helps the developer build an extensive enterprise-grade application. It can use hierarchical-based modeling constructs and encode the modeling workflow using some traditional programming languages like SQL, Java, R and so forth. This combination of workflow constructs would enable the developer to use hierarchical-based workflow for more straightforward tasks and fill up the remaining modeling workflow requirements using code technologies.

Conformity to standard: BPMN2.0 is standardized by ISO/IEC 19510^{****}. The International Standard defines four conformance types: Process Modeling Conformance, Process Execution Conformance, BPEL Process Execution Conformance,

****<https://www.omg.org/spec/BPMN/ISO/19510/PDF>

and Choreography Modeling Conformance. In contrast to BPMN, all the considered LCDPs are vendor-locked development platforms. Therefore, these LCDPs have their proprietary workflow mechanisms, although some of the LCDPs, such as OutSystems, and Mendix, have motivated their workflow mechanisms from the BPMN modeling construct standard.

Granularity: BPMN models can be modularized by using swimlanes that are divided into pools and lanes. Moreover, a BPMN model can be granularized according to three different modeling details that are useful for various stakeholders: (i) defining a process flowchart, including the overall activities and high-level decision conditions, is helpful for the end-user to understand; (ii) describing the whole process extensively on different roles of process, data, information and so forth may be beneficial for developers; (iii) explaining the modeling flowchart that comprises sufficient information such that the process may be defined for analysis and simulation. This detailed flowchart can then be executed to develop some code. The analyst uses this level of detail and is further handed over to the developer for proper execution in compliance with the detailed process model.

Compared to BPMN, LCDPs have separated workspaces to specify domain (data) models and for modeling user interfaces and processes. Also, some LCDPs such as Microsoft Powerapps, Amazon Honeycode, and Google Appsheet may support an integrated tool to process models within the user interface that can manipulate the data model designed for an application. Microsoft PowerApps does support OCL-based scripting inside the pages of the application along with the external usage of the Power Automate platform to granularize the process model from different pages of the application. The granularization in an LCDP can be divided into three parts:

- *Separation of process participants:* The entire process is built across different organizations; then, it is difficult to maintain consistency as there will be interoperability and authentication issues. For example, a company's recruitment process may require payment from an external payment gateway service done by the applicant. This means that the modeling construct built in a particular LCDP has to handle both of these processes within their platforms. For instance, the import module or the API-enabled feature in Mendix is used to interoperate with external services, which expert developers use to handle the processes of two distinct participants.
- *Separation of the roles of process participants:* The process built in an organization to support multiple stakeholders needs to be authenticated. Different user roles such as analysts, developers, users and so forth need to be filtered according to their roles and hierarchy within that organization. Such roles must be defined within the modeling construct of an LCDP. For example, the user roles in Thinkwise allow different organizational stakeholders to access a specific part of the application according to their defined roles.
- *Separation of processes, data, and user-interfaces:* Although the process modeling, data handling, and user-interface building can be overlapped to develop an application, it gives a more precise overview of the application if these three parts are separated. This allows the application to be agile with minimum coupling across these three parts allowing the developer to test and deploy the application. For example, OutSystems supports the separation of process, data, and user interface, which increases the application's maintainability as it could easily manage the consistency available in these three parts of the application development.

Executability and orchestability: BPMN Process Execution Conformance type is mandated in the BPMN execution tool that interprets models to the operational semantics. This tool supports the BPMN mapping to WS-BPEL (Web Service-Business Process Execution Language).¹⁰ BPMN Choreography Conformance is also implemented in BPMN packages. The package includes BPMN core elements, choreography diagrams, and collaboration diagrams that include pools and message flow. Since LCDPs are cloud-based application development platforms that mainly use models to develop an application, the process modeling constructs within an LCDP are executed either by code generation or by directly compiling models at run time.⁴⁰ Therefore, all the process modeling workflows within an LCDP are executed according to their execution engine. For example, some code-generated LCDPs are OutSystems, Thinkwise, Nintex Workflow and so forth while some LCDPs having compiled models to be executed at run-time are Mendix, Zoho Creator, Microsoft PowerApps, Salesforce Lightning, Amazon Honeycode, and Google Appsheet.

Measurability: Some aspects of BPMN2.0, such as graphical notations or metamodel quality, cannot be compared thoroughly with measurable quantities. However, automation and execution of BPMN models can be measured with a mostly one-to-one mapping of elements to their corresponding code. Therefore, BPMN supports partial measurability. For LCDPs, as all the process models must be either code generated or directly executed at run-time, each modeling element can be measured in terms of memory space and execution time. Therefore, process modeling can be measurable regarding execution time and the amount of required memory.

Business rules: BPMN supports a business rules engine that takes input from a process and produces the intermediate and final output. For example, a specified task gets executed by a business process engine. In LCDPs, inputs from multiple pages (such as forms or reports) are given to the process modeling used in developing an application. Also, multiple outcomes or tasks (such as scheduling tasks, approval tasks, etc.) can be executed using the designed process model within an LCDP. Therefore, all LCDPs support business rules.

Supporting tools: Some of the most used tools that support BPMN are Drawio, Microsoft Visio, Lucidchart, Visual Paradigm, SmartDraw and so forth. All the vendor-locked LCDPs support their built-in modeling tool.

Validation in real environments: Many vendors support the validation tools for the BPMN models, for example, viadee Process Application Validator in Camunda^{§§§§}, BPMN validation tool in WebRatio^{¶¶¶¶} and so forth. Also, another application named bpmnlink^{####} validates the BPMN diagrams based on configurable rules. These tools validate the syntactically BPMN modeling elements using configurable rules developed by programming languages such as Java. LCDPs support complete testing environments where developers can test applications and validate them. As the LCDP is either code generated or directly compiled on models, the validator is built using the concepts of model-driven engineering.³

With the listed quality standard/criteria of modeling languages mentioned above, different supported modeling constructs within a LCDP are considered. These considered modeling workflow guides the low-code application developer to choose a particular process modeling mechanism that is best suited for the application to be built.

Another quality criterion to consider when selecting LCDPs is the availability of automation of process modeling through Artificial Intelligence (by using Robotic Process Automation⁴¹) and machine learning techniques that can predict and suggest the following probable best action(s) based on the previous workflow used in a similar application. If a process construct in an LCDP builds a process workflow, the construct may suggest the next probable step(s) based on the learning from similar workflows. For instance, the OutSystems AI assistant works through the business workflow, predicts the next step, and recommends the best option available to configure and adapt a business logic according to the appropriate context^{||||||}. Similarly, in Microsoft PowerApps and Power Automate, AI Builder is used to help businesses to use AI through a point-and-click experience by building a custom model or choosing a pre-built model to train it and then using insights from that AI model^{*****}. These features go in the direction of intelligent modeling assistants.⁴²

6.1 | Threats to validity

In this section, we discuss the threats to validity by dividing them in intrinsic and extrinsic.

6.1.1 | Intrinsic threats

Some aspects might represent threats while evaluating data and business flows of different LCDPs. The free version of the considered LCDPs is used throughout the study. This means that some features, such as using APIs, collaboration features and so forth are not experimented with in handling external data sources. The study elaborates on the accessible version of the LCDPs, which focuses primarily on the direct use of external data without much coding or using different APIs. Also, there is no direct correspondence between the functionality of BPMN core elements and the LCDPs elements. To tackle this threat, BPMN elements' functionality is partially matched with the LCDPs elements. For example, the data model in Mendix handles the creation of data and their associated connecting objects with one another in a graphical manner. This data handling can only be represented as a separate entity apart from their connecting objects within the same or different swimlanes used in the BPMN2.0 specification.

§§§§ <https://camunda.com/blog/2017/10/viadeeprocessapplicationvalidator/>

¶¶¶¶ <https://methodandstyle.com/bpmn-validation-tool-improved-by-webratio/>

<https://github.com/bpmn-io/bpmnlint>

|||||| <https://www.outsystems.com/ai/>

***** <https://docs.microsoft.com/en-us/ai-builder/overview>

6.1.2 | Extrinsic threats

Some extrinsic factors that can threaten understanding the different kinds of data and business flow apart from the discussed LCDPs are as follows. The study focuses on 8 LCDPs only, and many other LCDPs, such as Kissflow, ZappDev, Nintex and so forth are not considered. However, we decided to focus the analysis by considering leading platforms that are niche players according to the Gartner report.⁵

7 | CONCLUSION & FUTURE WORKS

Business process management in application development plays a crucial role in improving business performance. To make the task agile and lean, the developer must create workflows involving several services. In this article, we described the usage of modeling frameworks that includes various process modeling along with the data-handling capabilities of LCDPs. The modeling constructs of various LCDPs are discussed with respect to those of BPMN. These modeling constructs are categorized, and the execution of the modeling constructs of different LCDPs is explained with the help of an illustrative case study.

In the future, we plan to continue analyzing existing LCDPs by also considering usability characteristics. The final goal is to conceive a comprehensive comparative framework supporting users in selecting LCDPs in an educated manner by covering both functional and extra-functional features.

ACKNOWLEDGMENTS

This research is supported by the Lowcomote Training Network,⁴³ which has received funding from the European Union's Horizon 2020 Research and Innovation Programme under the Marie Skłodowska-Curie Grant agreement no. 813884.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available on request from the corresponding author. The data are not publicly available due to privacy or ethical restrictions.

ORCID

Apurvanand Sahay  <https://orcid.org/0000-0003-3695-8598>

REFERENCES

1. Ruscio DD, Kolovos DS, de Lara J, Pierantonio A, Tisi M, Wimmer M. Low-code development and model-driven engineering: two sides of the same coin? *Softw Syst Model*. 2022;21(2):437-446. doi:10.1007/s10270-021-00970-2
2. Waszkowski R. Low-code platform for automating business processes in manufacturing. *IFAC-PapersOnLine*. 2019;52(10):376-381.
3. Sahay A, Indamutsa A, Di Ruscio D, Pierantonio A. Supporting the understanding and comparison of low-code development platforms. Proceedings of the 2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA); 2020:171-178; IEEE.
4. Georgakopoulos D, Hornick M, Sheth A. An overview of workflow management: from process modeling to workflow automation infrastructure. *Distrib Parallel Databases*. 1995;3(2):119-153.
5. Vincent P, Natis Y, Iijima K, et al. Magic quadrant for enterprise low-code application platforms. Gartner report; 2020.
6. Allweyer T. BPMN 2.0: introduction to the standard for business process modeling. BoD-books on demand; 2016.
7. Recker J, Rosemann M, Indulska M, Green P. Business process modeling-a comparative analysis. *J Assoc Inf Syst*. 2009;04:10.
8. Reisig W. *Petri Nets: An Introduction*. Vol 4. Springer Science & Business Media; 2012.
9. White SA. Introduction to BPMN; Vol. 2, 2004; IBM Cooperation.
10. Ouyang C, Dumas M, Aalst WMVD, Hofstede AHT, Mendling J. From business process models to process-oriented software systems. *ACM Trans Softw Eng Methodol (TOSEM)*. 2009;19(1):1-37.
11. MacDonell S, Shepperd M, Kitchenham B, Mendes E. How reliable are systematic reviews in empirical software engineering? *IEEE Trans Softw Eng*. 2010;36(5):676-687.
12. Bendraou R, Jezequel JM, Gervais MP, Blanc X. A comparison of six UML-based languages for software process modeling. *IEEE Trans Softw Eng*. 2010;36(5):662-675.
13. Wise A, Cass AG, Lerner BS, McCall EK, Osterweil LJ, Sutton SM. Using little-JIL to coordinate agents in software engineering. Proceedings ASE 2000 15th IEEE International Conference on Automated Software Engineering; 2000:155-163.
14. García-García JA, Enríquez JG, Domínguez-Mayo F. Characterizing and evaluating the quality of software process modeling language: comparison of ten representative model-based languages. *Comput Stand Interf*. 2019;63:52-66.

15. Portela C, Vasconcelos A, Sinimbú A, et al. A comparative analysis between BPMN and SPEM modeling standards in the software processes context. *J Softw Eng Appl*. 2012;05(05):330-339.
16. Guizani K, Ghannouchi SA. An approach for selecting a business process modeling language that best meets the requirements of a modeler. *Proc Comput Sci*. 2021;181:843-851.
17. Hassen MB, Turki M, Gargouri F. Choosing a sensitive business process modeling formalism for knowledge identification. *Proc Comput Sci*. 2016;100:1002-1015.
18. Wang W, Ding H, Dong J, Ren C. A comparison of business process modeling methods. Proceedings of the 2006 IEEE International Conference on Service Operations and Logistics, and Informatics; 2006:1136-1141; IEEE.
19. Andrews T, Curbera F, Dholakia H, et al. Business process execution language for web services. Version; 2003.
20. Force URT. OMG unified modeling language: superstructure; 2010; Object Management Group (OMG).
21. from Workflow Process RD, from Schema RPE, Element AS. Workflow process definition interface—XML process definition language. Lighthouse point (FL): workflow management coalition, (WFMCTC-1025); 2005.
22. Petri CA, Reisig W. Petri net. *Scholarpedia*. 2008;3(4):6477.
23. Dorador J, Young RI. Application of IDEF0, IDEF3 and UML methodologies in the creation of information models. *Int J Comput Integr Manuf*. 2000;13(5):430-445.
24. Pereira JL, Silva D. Business process modeling languages: a comparative framework. In Bahsoon R, (ed.), *New Advances in Information Systems and Technologies*. Springer; 2016:619-628.
25. Kelemen ZD, Kusters R, Trienekens J, Balla K. Selecting a process modeling language for process based unification of multiple standards and models. Technical report TR201304, Budapest; 2013.
26. Birkmeier D, Overhage S. Is BPMN really first choice in joint architecture development? An empirical study on the usability of BPMN and UML activity diagrams for business users. Proceedings of the International Conference on the Quality of Software Architectures; 2010:119-134; Springer.
27. Gartner. Gartner forecasts. Gartner; 2022. Accessed October 28, 2022. <https://www.gartner.com/en/newsroom/press-releases/2021-02-15-gartner-forecasts-worldwide-low-code-development-technologies-market-to-grow-23-percent-in-2021>
28. OutSystems. OutSystem platform business process. OutSystems Community; 2022. Accessed November 09, 2022.
29. Martins R, Caldeira F, Sá F, Abbasi M, Martins P. An overview on how to develop a low-code application using OutSystems. Proceedings of the 2020 International Conference on Smart Technologies in Computing, Electrical and Electronics (ICSTCEE); 2020:395-401; IEEE.
30. Mendix. Mendix Microflow. Mendix docs; 2022. Accessed November 09, 2022. <https://docs.mendix.com/refguide/microflows/>
31. Creator Z. Zoho creator deluge script. Zoho creator 2022. Accessed November 09, 2022. <https://www.zoho.com/creator/help/images/intro-to-deluge-workflows.pdf>
32. Cabot J, Gogolla M. Object constraint language (OCL): a definitive guide. In: Bernardo M, Cortellessa V, Pierantonio A, eds. *International School on Formal Methods for the Design of Computer, Communication and Software Systems*. Springer; 2012:58-90. doi:10.1007/978-3-642-30982-3_3
33. PowerApps M. Microsoft PowerApps flow. PowerApps; 2022. Accessed November 09, 2022. <https://docs.microsoft.com/en-us/powerapps/user/use-flows>
34. Keel J. *Lightning Process Builder and Visual Workflow*. Springer; 2016 [Salesforce.com](https://www.salesforce.com)
35. Lightning S. *Salesforce Lightning Flow*. Salesforce Inc; 2022 <https://www.salesforce.com/in/products/platform/solutions/automate-business-processes/>. Accessed November 09, 2022
36. Thinkwise. *Thinkwise Process Flow*. Thinkwise; 2022 https://docs.thinkwisesoftware.com/docs/2020.1/sf/process_flows.html. Accessed November 09, 2022
37. Appsheet G. *Google Appsheet Actions*. Appsheet Help; 2022 <https://help.appsheet.com/en/articles/953637-actions-theessentials>. Accessed November 09, 2022
38. HoneyCode A. *Amazon Honeycode Capabilities*. Honeycode; 2022 <https://www.honeycode.aws/features>. Accessed January 03, 2021
39. Corradini F, Ferrari A, Fornari F, et al. A guidelines framework for understandable BPMN models. *Data Knowl Eng*. 2018;113:129-154.
40. Song H, Huang G, Chauvel F, Sun Y. Applying MDE tools at runtime: experiments upon runtime models. Proceedings of the 5th International Workshop on Models at Run Time; 2010:16.
41. Aguirre S, Rodriguez A. Automation of a business process using robotic process automation (RPA): a case study. Proceedings of the Workshop on Engineering Applications; 2017:65-71; Springer.
42. Mussbacher G, Combemale B, Kienzle J, et al. Opportunities in intelligent modeling assistance. *Softw Syst Model*. 2020;19(5):1045-1053.
43. Tisi M, Mottu J, Kolovos DS, et al. Lowcomote: training the next generation of experts in scalable low-code engineering platforms. Co-Located Events Joint Proceedings with Software Technologies: Applications and Foundations, STAF 2019, Vol. 2405 of CEUR Workshop Proceedings; 2019:73-78. [CEUR-WS.org](https://ceur-ws.org).

How to cite this article: Sahay A, Di Ruscio D, Iovino L, Pierantonio A. Analyzing business process management capabilities of low-code development platforms. *Softw Pract Exper*. 2023;53(4):1036-1060. doi: 10.1002/spe.3177