sustainability

MDPI

Article

Visibility Matrix: Efficient User Interface Modelling for Low-Code Development Platforms

Robert Waszkowski [1,*] and Grzegorz Bocewicz [2]

[1] Faculty of Cybernetics, Military University of Technology, 00-908 Warszawa, Poland
[2] Faculty of Electronics and Computer Science, Koszalin University of Technology, 75-453 Koszalin, Poland; grzegorz.bocewicz@tu.koszalin.pl
* Correspondence: robert.waszkowski@wat.edu.pl; Tel.: +48-602-682-683

Abstract: In this paper, we introduce the idea of the 'visibility matrix' for automated data entry form generation in low-code development platforms. We then focus on the problem of software development productivity in the area of automated software generation as the main factor of the Industry 4.0 concept in the area of business information. In our study, two different approaches to user interface development in a business process management low-code platform were evaluated. The first, the multi-form model, assumes that input forms are prepared separately for each user task in the business process being automated. The second approach, the single-form model, assumes that there is one global input form for every task in the business process. Since users have access to different data in different process tasks, it is necessary to prepare the visibility matrix to define which data are relevant to which tasks. The experiments presented in this paper help to answer the following question: which approach yields better results in terms of productivity, which is measured as costs and time required to prepare the application? Several dozen real business processes were analysed to examine the properties of their visibility matrix. Additionally, the real project team members were evaluated to determine their productivity. Then, the productivity parameters were calculated for real business processes and real project teams. The results show which approach is better suited for real-world business process development.

Keywords: design tools and techniques; productivity; software engineering; user interfaces; tools; visibility matrix; Aurea BPM; business process management; low-code development platform

Citation: Waszkowski, R.; Bocewicz, G. Visibility Matrix: Efficient User Interface Modelling for Low-Code Development Platforms. *Sustainability* **2022**, *14*, 8103. https://doi.org/10.3390/su14138103

Academic Editors: Marta Rossi, Beata Mrugalska, Marco Mandolini, Paolo Cicconi, Yingjie Yang, Alessandra Papetti and Paolo Salini

Received: 15 May 2022
Accepted: 29 June 2022
Published: 2 July 2022

1. Introduction

According to the Plex's seventh annual "State of the Smart" manufacturing report, 80% of manufacturers consider smart manufacturing to be critical to their future success. Smart manufacturing involves the use of an interconnected network of equipment, machinery, and processes. Integrating smart technologies such as IoT and other Lean Manufacturing and Industry 4.0 initiatives typically requires the use of specialised applications and platforms. Due to the costs of developing new applications, smaller manufacturers are often the last to benefit from custom purpose-built software solutions. Low-code development platforms (LCDP) provide an alternative solution [1]. The advantages of an LCDP for manufacturing include shorter development life cycles, faster implementation of new processes, more efficient workflows, and less need for specialised IT.

A low-code platform is a set of tools for programmers and non-programmers. It enables quick generation and delivery of business applications with minimum effort to write in a coding language and requires the least possible effort for the installation and configuration of environments, as well as training and implementation. With a rapidly growing number of companies, the use of low-code solutions can be a significant step forward in creating essential business applications.

In 2017, Joe McKendrick published a report on the development of applications by non-programmers, titled "The Rise of the Empowered Citizen Developer: Is IT Possible

Without IT" [2]. This report shows that the main problem for business departments is the long waiting time for delivery of a business application to end-users and the long waiting time for requested data and reports. This leads to situations in which so-called 'citizen development' (software developing by non-programmers) takes place in business departments on its own, and with the use of common office platforms. More serious and responsible engagement of people who create such applications will certainly lead to the faster and better creation of business applications, especially if they are equipped with appropriate tools [3,4]. This leads us to the rise of new software development tools: low-code development platforms.

This paper will specifically focus on building applications to support the execution of business process tasks. In this regard, authors will focus on one aspect of low-code development, i.e., data entry form preparation in business process user interface. Two different approaches will be evaluated: the multi-form model and the single-form model.

The multi-form model assumes that input forms will be prepared separately for each user task in the business process being automated. The single-form model assumes that there will be one global input form for every task in the business process.

In real business processes, there are many tasks, each of which has its own electronic form with which to interact with users. Therefore, all the work required for form preparation must be replicated for each task that presents data for reading or editing by task executors.

When it comes to system implementation, a dilemma arises: whether to use one central form for the entire business process or use separate forms for each task. The use of one common form to manipulate the process data (single-form development model) seems to be an extremely desirable approach due to the smaller workload in terms of modelling and coding. In this case, all the work required for form preparation is performed only once, and not as many times as the number of tasks in the process (usually from five to even twenty). The question arises: is it possible to use the single-form development model without compromising the quality and completeness of the final solution, an IT system for supporting business processes?

The single-form development model allows one to define a global data model for the entire business process and develop only one form. The workload is much less than for the multi-form model, but each task must be handled by the same form. This makes it difficult to prepare specific forms for individual tasks.

This problem can be mitigated by defining the 'data visibility matrix' for each business process. It contains information about which variable from the process data is to be used in which task.

In further considerations, both approaches to building data entry forms will be compared to determine which approach is more effective.

In this context, the following structure of the article is proposed. In Section 2, the state of the art, i.e., an overview of existing approaches and models related to increasing the efficiency of software development, is presented. Section 3 includes the formulation of the research problem, the related motivational example, and the sufficient conditions, the fulfilment of which guarantees a shorter time spent implementing the business process (thanks to the single-form model approach). Section 4 contains the results of the quantitative and qualitative experiments undertaken. In Section 5, a discussion of the results obtained is presented.

2. State of the Art

The term 'low-code development' was first used by Clay Richardson in 2014 in a report for Forrester Research entitled: "New Development Platforms Emerge For Customer-Facing Applications" [5]. Low-code platforms require less coding knowledge. That is why practically everyone, including business ('citizen') developers, is able to use a low-code development platform. This feature is extremely important in the current times as the market faces a serious lack of skilled software engineers.

Although the concept of low-code development is quite young, there are many platforms in the market supporting this idea [6]. There is a lot of discussion about the usefulness of the platforms, and there are both supporters and opponents, including Abdullah Al Alamin [7], Jason Bloomberg [8], Marcus Woo [9], and Myles F. Cio [10]. The fact is that low-code platforms are evolving, and even stronger development is planned in the future. They are used in many branches of industry and trade as well as in administration units, as described by Arpan Bansal [11], Raquel Sanchis [1], and Cecilie Ness [12].

Along with the development of technology, scientific articles that assess selected aspects of its technical functioning have emerged, e.g., Ana Nunes Alonso [13], Ilene Wolff [14], and Michele Missikoff [15]. Some of them raise issues related to software development productivity [16], which is the main problem considered in this article.

However, among these publications there are no articles that are strictly related to the problem of software development productivity, especially in the area of automated data entry form preparation in LCDPs. The literature separately addresses the issues of improving productivity and building user interfaces. There are practically no references to both together in the context of low-code platforms.

Various methods of generating user interfaces in an automated manner and based on models are discussed in the works of Claudia Veronica Serey Guerrero and Bernardo Lula [17], Pierre A. Akiki, Arosha K. Bandara and Yijun Yu [18], Lassaad Ben Ammar [19], Gordana Milosavljević, Dragan Ivanović, Dušan Surla and Branko Milosavljević [20], and D.I. Heimann [21]. A more practical approach, supported by examples of applications, is presented by Christophe Kolski, Jean Vanderdonckt, and Henry Lieberman [22–27], Jano van Hemert, Jos Koetsier, Livia Torterolo, Ivan Porro, Maurizio Melato, and Roberto Barbera [28], and P. Bocciarelli [29].

The problem of user interface generation is reviewed in the work of Mary Ann [30], and the formal point of view is presented by Jürgen Engel, Christian Märtin, Christian Herdin, and Peter Forbrig [31]. The problem of appliance methods is described by G. Dodig-Crnkovic et al. in [32].

Productivity improvement is also discussed in many studies. Some of them are strictly related to the productivity of creating user interfaces. The book section "Evaluation of Model-Based User Interface Development Approaches" in "Human-Computer Interaction. Theories, Methods, and Tools" by Jürgen Engel, Christian Herdin, and Christian Märtin [33] deserves special attention. The authors propose the PaMGIS framework developed at Augsburg University of Applied Sciences to support user interface designers without profound software development skills to specify the diverse models, which allow for at least semi-automated generation of user interface source code. They also evaluated the existing model-based user interface development frameworks in order to elicit new ideas to improve the applicability of their PaMGIS solution. Valeriya Gribova, in the book section "Methods for Decreasing Time and Effort during Development and Maintenance of Intellectual Software User Interfaces" in "Advanced Intelligent Computing Theories and Applications With Aspects of Theoretical and Methodological Issues" [34] proposes methods for automated development and maintenance of intellectual software user interfaces. She uses an ontology-based approach that intends to decrease effort and time for user interface development and maintenance. Gregory Alan Bolcer, in the article "User interface design assistance for large-scale software development" [35], addresses the specific design problems of style and integration consistency throughout the user interface development process and aids in the automated feedback and evaluation of a system's graphical user interface according to knowledge-based rules and project-specific design examples.

The literature describes issues similar to those covered in this article, i.e., software development productivity in the area of automated data entry form preparation in LCDPs (Table 1). However, due to the highly specific formulation of the problem and the highly focused scope of our research, the problem described in the paper should be treated as part of a larger whole that contributes to the overall productivity of the user interface. The

problem as such was not formulated by other scientists and was not solved. In this regard, our research is new and unique.

Table 1. State of the art.

Area of Contribution	References
General low-code development concept	[5,6]
Discussion on the usefulness of the LCDP platforms	[7–10]
Technical functioning aspects of LCDPs	[13–15]
Software development productivity in LCDPs	[16]
Automated user interface generation	[18–32]
Productivity improvement	[33–35]

3. Motivation and Research Problem

3.1. Motivation

Let us consider a situation where there is a business process (BP_1), as shown in Figure 1. The process includes five User Tasks (a_1–a_5). This kind of tasks means that a human performer executes the task with the use of a software application, i.e., data entry form. This means that five data entry forms (FR_1–FR_5) must be prepared in order to handle all process tasks (a_1–a_5).

However, the business process operates on a certain range of data (Table 2). These data can be read and modified by a user. To allow a user to operate on the data, the data entry forms must be prepared.

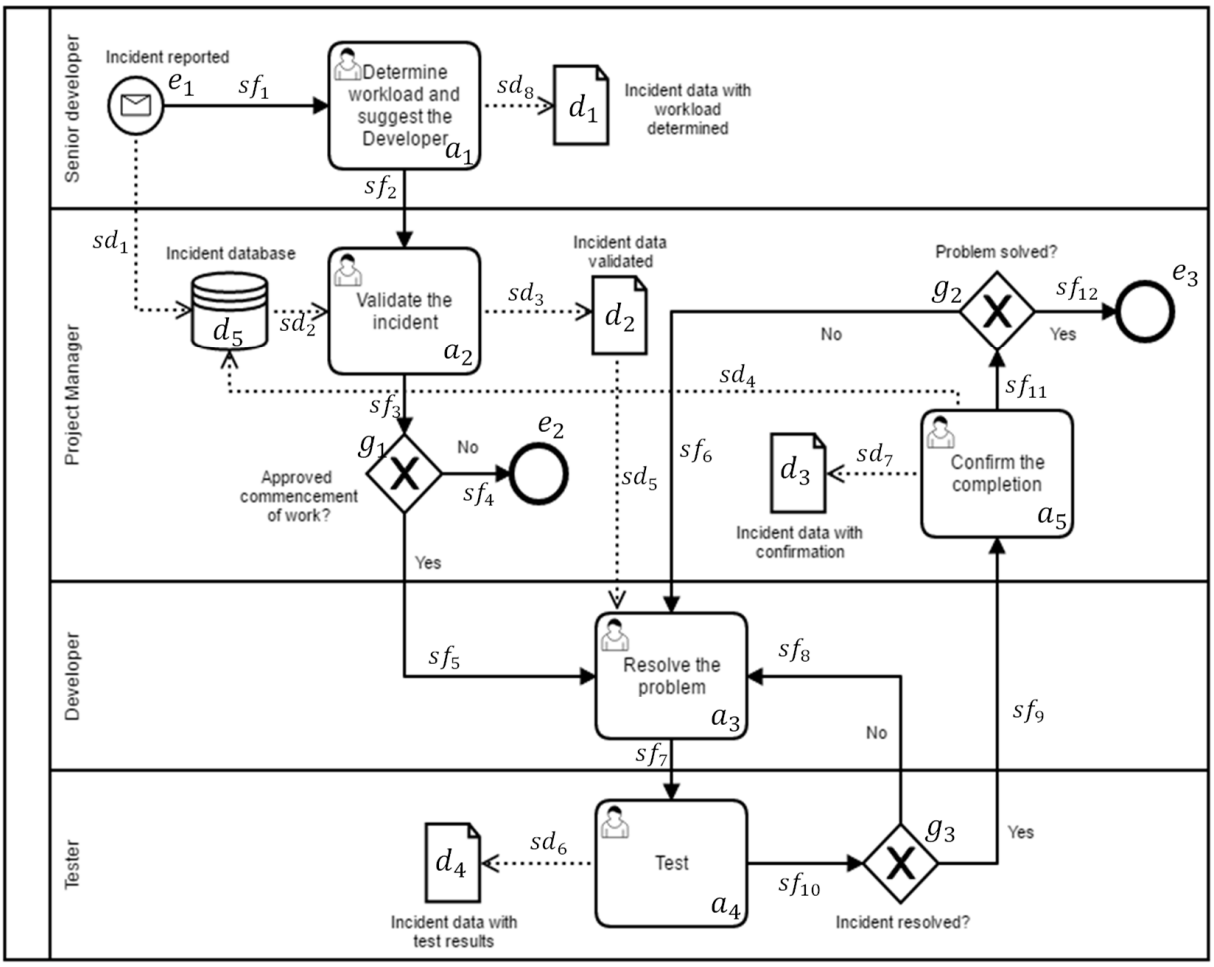
Table 2. Variables defined for the helpdesk business process (BP_1).

Variable	Data Type	Description
Subject	Character	The subject of the incident
Project	Character	The project that the incident relates to
Priority	Integer	The problem priority (1–9)
Incident date	Date	The date on which the incident occurred
Expected due date	Date	The expected date of solution delivery
Workload	Integer	The estimated workload of incident
Incident source	Character	The name of the person reporting the incident
Source email	Character	An email of the person reporting the incident
Developer assigned	Character	The developer assigned to resolve the problem
Type of incident	Enum	The type of incident
App version	Character	The current version of the application
Incident description	Character	A description of the incident (what is not working properly?)
Approved?	Boolean	An indicator whether the incident is approved for processing
Incident resolved?	Boolean	An indicator whether the incident is considered as resolved
Problem solved?	Boolean	An indicator whether the incident is completed

Let us consider that we use the single-form model for data entry form preparation. That means that a data entry form must be created for each task in the BP_1 business process.

Figure 2 presents the forms that should be created for tasks a_1 and a_2. As is evident, they only differ regarding one variable.

The construction of all the data entry forms (FR_1–FR_5) for the business process (BP_1) requires the necessity of building the same elements many times (e.g., re-construction of fields for variables x_1–x_{12}). After analysing the data processed by individual tasks, it can be seen that the tasks have a significant number of common variables (Table 3). In the case under consideration, twelve variables (x_1–x_{12}) are common to all five forms (a_1–a_5). This means that the use of the multi-form model approach to the construction of these forms entails a redundancy of work related to creating fields for these variables (the form items for each of these variables are created five times). In the considered case, the construction of forms for tasks a_1–a_5 of the process from Figure 1 requires the creation of 63 fields (form items).



Legend:

Name of task a_i - user task $a_i \in V_{FE}$

g_i - gateway $g_i \in V_{FG}$

e_i - event $e_i \in V_{FE}$

d_i - data object $d_i \in V_D$

sf_i - sequence flow edge $sf_i \in E_F$

sd_i - data association $sd_i \in E_D$

Figure 1. Sample helpdesk business process (BP_1). Source: own elaboration.

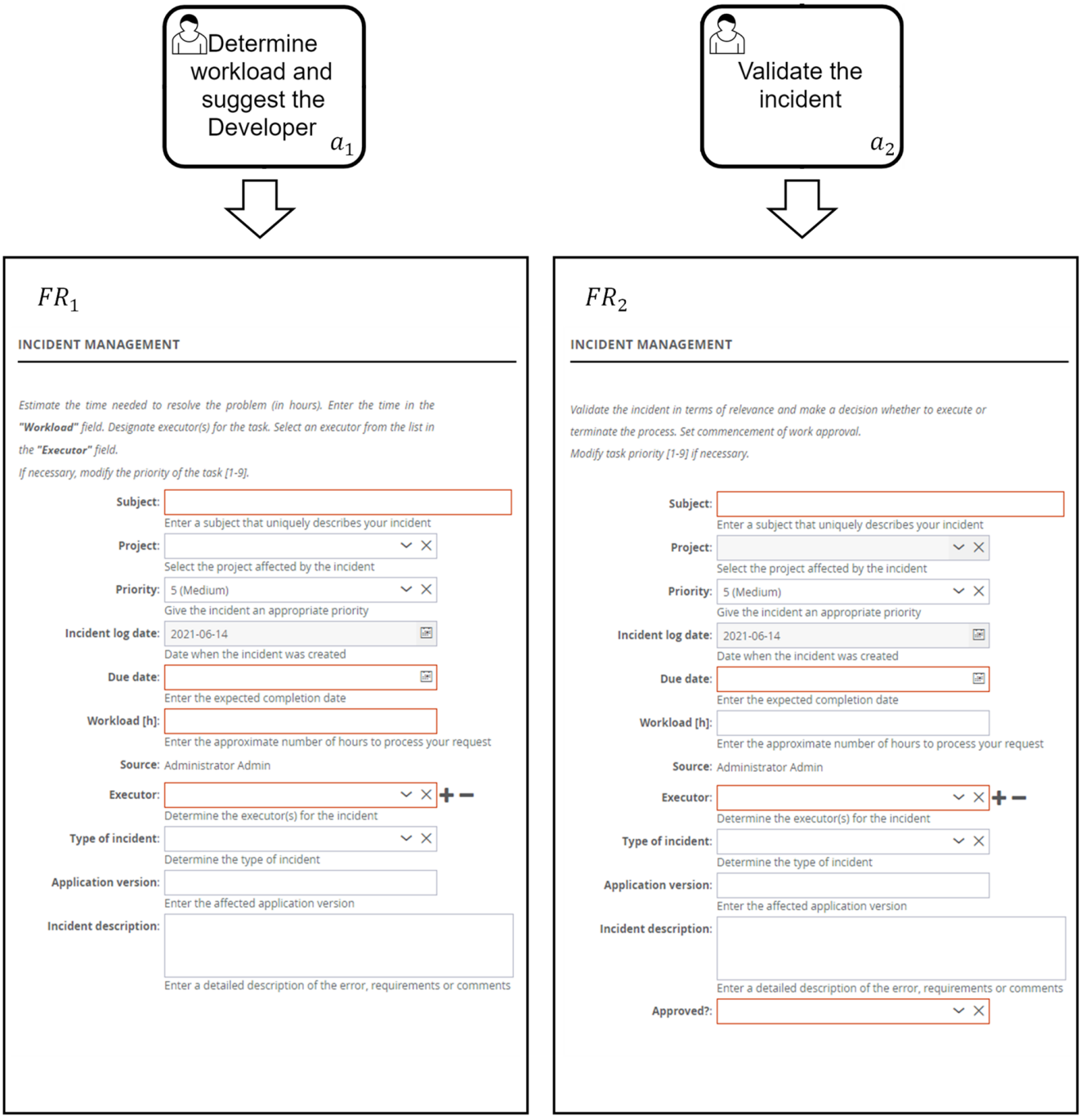
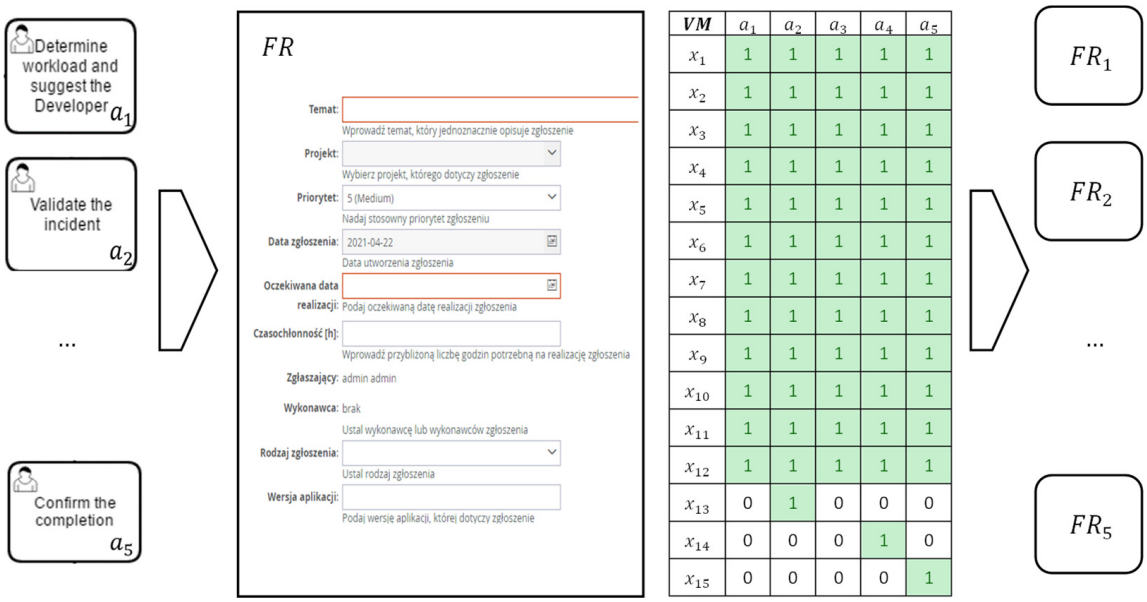


Figure 2. The idea of the multi-form model for the sample business process (2 forms out of 5 are shown). Source: own elaboration.

An alternative approach, the single-form model, is illustrated in Figure 3. In this approach, only one electronic form that serves all tasks is built (FM) and the so-called visibility matrix (VM) is prepared. The VM is the binary matrix that represent the binary relation between the variables used in a business process and the business process' tasks. It indicates which field is relevant to which task in terms of reading and modifying of process data by a business user (Table 3). Based on the VM, data entry forms can be generated automatically for each task. In this case, the construction of the FM form requires the creation of 15 fields and a VM of size 15×5. It should be noted that the time for building one form field (hereinafter denoted by the variable t_a) is much longer than the time for determining the value of a single cell of the VM (hereinafter denoted by the variable t_c).

This means that the use of the single-form model approach to generate task forms a_1–a_5 has almost four times less cost than in the case of using the multi-form model.



VM	a_1	a_2	a_3	a_4	a_5
x_1	1	1	1	1	1
x_2	1	1	1	1	1
x_3	1	1	1	1	1
x_4	1	1	1	1	1
x_5	1	1	1	1	1
x_6	1	1	1	1	1
x_7	1	1	1	1	1
x_8	1	1	1	1	1
x_9	1	1	1	1	1
x_{10}	1	1	1	1	1
x_{11}	1	1	1	1	1
x_{12}	1	1	1	1	1
x_{13}	0	1	0	0	0
x_{14}	0	0	0	1	0
x_{15}	0	0	0	0	1

Figure 3. The idea of the single-form model for the sample business process. Source: own elaboration.

Table 3 shows the VM for the (BP_1) business process. The 'x' sign in the matrix cell indicates that the given variable is used in the given business process task.

Table 3. The use of variables in different tasks (visibility matrix for the (BP_1) business process).

		Project Tasks				
		a_1	a_2	a_3	a_4	a_5
	Variable	**Determine Workload . . .**	**Validate the Incident**	**Resolve the Problem**	**Test**	**Confirm the Completion**
x_1	Subject	x	x	x	x	x
x_2	Project	x	x	x	x	x
x_3	Priority	x	x	x	x	x
x_4	Incident date	x	x	x	x	x
x_5	Expected due date	x	x	x	x	x
x_6	Workload	x	x	x	x	x
x_7	Incident source	x	x	x	x	x
x_8	Source email	x	x	x	x	x
x_9	Developer assigned	x	x	x	x	x
x_{10}	Type of incident	x	x	x	x	x
x_{11}	App version	x	x	x	x	x
x_{12}	Incident description	x	x	x	x	x
x_{13}	Approved?		x			
x_{14}	Incident resolved?				x	
x_{15}	Problem solved?					x

The above example shows that the construction of the data entry forms used in the given business process (BP_1) can be done in many ways (multi-form model or single-form model), which, due to the redundancy of work, result in different costs of their implementation (and thus a different working time). In a situation where many tasks of a business process share the same resources (variables in Table 3), it may turn out that the construction of a common form (FM) and VM that enables its parameterisation for individual tasks of the process (BP_1) may be less expensive than the traditional approach commonly used by developers of building a separate form for each task (multi-form model).

It should be emphasised that the benefit of use in the single-form model approach occurs due to the presence of the tasks in the process that share multiple variables; in other words, Table 2 is 'dense'. Practice shows that this approach becomes ineffective for processes whose tasks share very few variables. In this approach, the problem considered in the current paper concerns the evaluation of the costs of applying the presented approaches. In other words, an answer is sought to the question: what features of the business process (and its tasks) determine the advantage (lower implementation costs) of the single-form model approach over the multi-form model approach (or vice versa)?

3.2. Statement of Research Problem

In general, a business process (see Figure 1) can be represented as a directed graph defined as a pair:

$$BP = (V, E) \tag{1}$$

where, for BPMN notation:

$V = V_F \cup V_D$ means the set of vertices represent:

- flow nodes: $V_F = V_{FE} \cup V_{FA} \cup V_{FG}$ containing events (V_{FE}), activities/user tasks (V_{FA}) and gateways (V_{FG}). For example, the process shown in Figure 1 includes 3 events $V_{FE} = \{e_1, e_2, e_3\}$, 5 user tasks $V_{FA} = \{a_1, a_2, \ldots, a_5\}$ and 3 gateways $V_{FG} = \{g_1, g_2, g_3\}$,
- data objects V_D. For example, the process shown in Figure 1 includes 5 data objects $V_D = \{d_1, d_2, \ldots, d_5\}$. $E \subseteq V \times V$ is the set of edges. It consists of:
- set of edges representing sequence flows E_F. For example, the business process (BP_1) shown in Figure 1 includes 12 sequence flows $E_F = \{sf_1, sf_2, \ldots, sf_{12}\}$,
- set of edges representing message flows E_M,
- set of edges representing data associations E_D. For example, the business process (BP_1) shown in Figure 1 includes 12 edges of the data association type $E_D = \{sd_1, sd_2, \ldots, sd_8\}$,
- set of edges representing conversation links E_L.

The implementation of the BP business process defined in this way includes a number of stages, among which the construction of forms for handling tasks from the V_{FA} set deserves special attention. The BPMN notation does not provide for formal data models. Thus, data related to specific tasks V_{FA} can be modeled in many different ways [36,37].

Let us simply assume that each task $a_i \in V_{FA}$ of the BP process is related to a finite set of variables $X(a_i) = X_i$. Access to variables from the X_i set is usually implemented through the set of GUI forms. The form FR_i dedicated to task $a_i \in V_{FA}$ of the BP process allows the user to specify the values of the X_i variables needed to accomplish this task. The size f_i of the form FR_i (number of fields in the form) is determined by the number of X_i variables and equals $f_i = |X_i|$.

Programmers and citizen developers can use two approaches to modelling data usage in the GUI:

1. **multi-form model**, in which a separate form (FR_i) is built for each task $a_i \in V_{FA}$ of the BP process. In this case, the number of forms is equal to the number of tasks in the process: $n = |V_{FA}|$,
2. **single-form model**, in which all the forms FR_i for each task $a_i \in V_{FA}$ are automatically generated from a so-called master form (FM) together with the VM: $FR_i = f(FM, VM, i)$.

Figure 4 presents the concept of constructing forms for BP process tasks in accordance with the multi-form model approach. This approach assumes that programmers or citizen developers construct the set of separate forms (FR_i) for each task $a_i \in V_{FA}$. As shown in the example in the previous section (Figure 2), this approach involves redundant work in situations where tasks use the same variables (which is often the case in practice). Avoiding redundant work is possible thanks to the use of a single-form model.

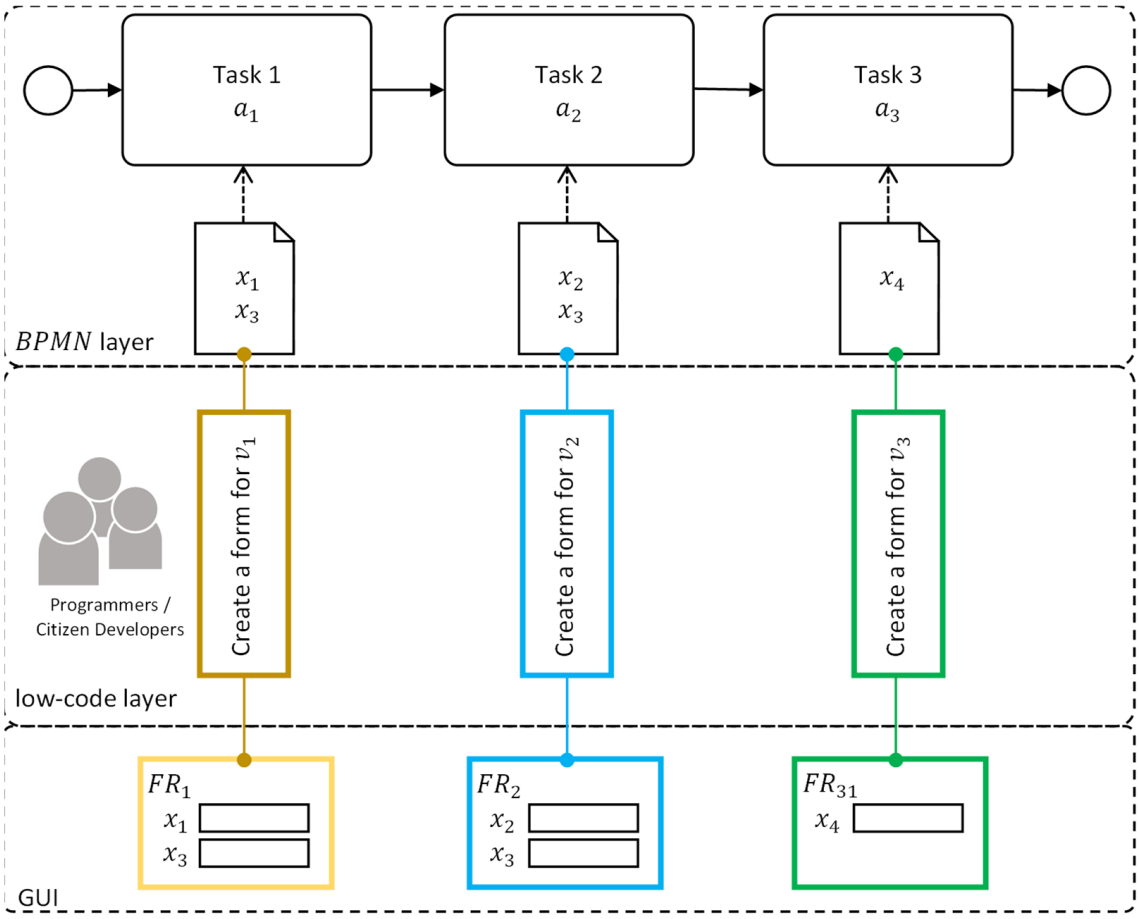


Figure 4. The principle of form construction in a multi-form model approach. Source: own elaboration.

The master form (FM) used in the single-form model approach utilises all the BP process variables: $\cup_{i=1}^{n} X_i$ cardinality of which equals to $m = \left|\cup_{i=1}^{n} X_i\right|$. While VM is the matrix $\left[vm_{i,j}\right]_{m \times n}$, elements $vm_{i,j} \in \{0,1\}$ determine the visibility of the variable j in the FR_i form of the task $a_i \in V_{FA}$:

$$VM = \left[vm_{i,j}\right]_{m \times n} \tag{2}$$

where $vm_{i,j} \in \{0,1\}$ equals 1 when variable j is necessary in the FR_i form of the task $a_i \in V_{FA}$ and equals 0 (zero) when variable j is not necessary in the FR_i form of the task $a_i \in V_{FA}$.

The principle of construction of the FR_i form, based on FM, VM, is illustrated in Figure 5. This approach assumes that for a given process (BP), programmers or citizen developers are responsible for building the FM and VM. The forms FR_i (GUI layer), designed to handle individual tasks, are generated automatically from the FM based on the data contained in the VM. In other words, each of the FR_i forms are obtained from FM by hiding unnecessary fields.

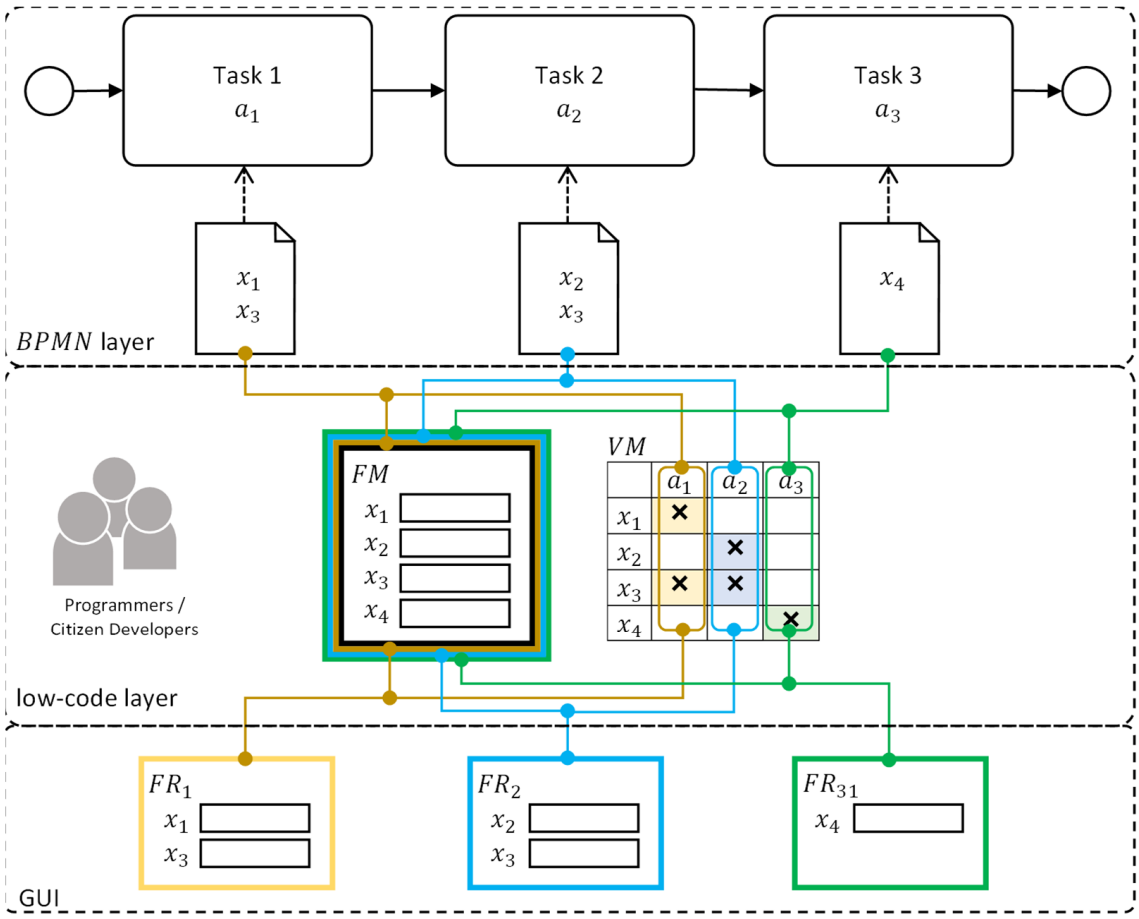


Figure 5. The principle of form construction in a single-form model approach based on the master form (FM) and the visibility matrix (VM). Source: own elaboration.

3.3. Sufficient Condition

Answering this question requires formulating conditions, the fulfilment of which guarantees lower implementation costs using the single-form model approach ($K^C_{m,n} < K^R_{m,n}$). For this purpose, the following auxiliary variables are introduced, which characterise, inter alia, effectiveness of project teams.

- $TCA = t_c/t_a$: a coefficient that determines how many times the filling time of one visibility matrix cell is shorter than the construction time of one form field (it is usually the goal of the project team to keep the TCA as low as possible),
- $TBA = t_b/t_a$: a coefficient that determines how many times the form initialisation time is longer than the construction time of one form field,
- $D = \frac{\sum_{i=1}^{n} f_i}{m \times n}$: the density of the VM, $D \in [0, 1]$.

It is assumed that the cost $K^C_{m,n}$ of the implementation of the BP process in the single-form model approach consists of the time to prepare a blank form (t_b), the time to embed all the process variable fields into the form ($m \times t_a$), and the time to fill the whole VM ($m \times n \times t_c$):

$$K^C_{m,n} = t_b + m \times t_a + m \times n \times t_c \tag{3}$$

For example, assuming that the effectiveness of a developer implementing the BP_1 process shown in Figure 1 ($m = 15,\ n = 5$) is as follows,

- time to embed one variable field into the form: $t_a = 300$ [s] (5 min),
- time to prepare a blank form: $t_b = 600$ [s] (10 min),
- time to fill one cell of the VM: $t_c = 30$ [s],

The time needed to prepare the data entry forms for the tasks a_1–a_5 equals $K^C_{15,5} = 2042$ [h] (7350 s).

Similarly, the cost $K^R_{m,n}$ of the implementation of the BP process in the multi-form model approach consists of the time required to prepare n blank forms ($n \times t_b$) and the time required to embed the variable field into them $(f_1 + \cdots + f_n) \times t_a$:

$$K^R_{m,n} = n \times t_b + (f_1 + \cdots + f_n) \times t_a \tag{4}$$

Assuming the same developer efficiency parameters, the time of the developer's work related to the preparation of forms in accordance with the multi-form model is $K^R_{15,5} = 6083$ [h] (21,900 s). This means that the use of a single-form model in the case under consideration allows for a threefold reduction in the developer's working time. Therefore, the question arises, under which conditions is the use of a single-form model in user interface preparation still profitable (in the sense of a shorter delivery time $K^C_{m,n}$)?

The single-form model approach is preferable to the multi-form model approach when:

$$K^C_{m,n} < K^R_{m,n} \tag{5}$$

Considering (2) and (3), this inequality can be written as:

$$t_b + m \times t_a + m \times n \times t_c < n \times t_b + t_a \times \sum_{i=1}^{n} f_i \tag{6}$$

$$\sum_{i=1}^{n} f_i > \frac{m \times t_a + m \times n \times t_c + t_b \times (1-n)}{t_a} \tag{7}$$

Considering further that $TCA = t_c/t_a$, $TBA = t_b/t_a$, and $D = \frac{\sum_{i=1}^{n} f_i}{m \times n}$:

$$\sum_{i=1}^{n} f_i > m + m \times n \times TCA + (1-n) \times TBA \tag{8}$$

$$D \times m \times n > m + m \times n \times TCA + (1-n) \times TBA \tag{9}$$

$$D > \frac{1}{n} + TCA + \frac{(1-n) \times TBA}{m \times n} \tag{10}$$

$$D > DG \tag{11}$$

where $DG = \frac{1}{n} + TCA + \frac{(1-n) \times TBA}{m \times n}$ determines the limit value (minimum) of the density (D) of the VM for which condition (4) is met, i.e., the single-form model is less expensive than the multi-form model (cost-effective threshold density; DG). The above condition leads to the following theorem:

Theorem 1. *For given process BP (1), the inequality $K^C_{m,n} < K^R_{m,n}$ is satisfied if the inequality $D > DG$ is satisfied.*

Proof. The proof results directly from the argumentation (4)–(11). □

The presented theorem should be interpreted as follows. If the density D of the VM for the BP (1) business process is greater than DG (3), then its implementation is more beneficial (i.e., it requires less working time) with the use of the *single-form model* approach: $K^C_{m,n} < K^R_{m,n}$.

As already mentioned in the example given in Figure 1, the value of $K^R_{15,5}$ is greater than $K^C_{15,5}$. This is due to the fact that the density value $D = 0.84$ is greater than $DG = 0.19$; thus, the Theorem 1 condition is met.

The condition developed is essentially a criterion for applying one of the two modeling approaches. The choice of the best model depends on the density (D) of the VM (which depends on the BP process structure) and on DG, which reflects the project team's production capacity (represented by TCA and TBA coefficients). In this context, in addition

to assessing the profitability of a specific modeling approach, the introduced measures can be used to determine the changes to be made in the BP process structure or in the configuration of the project team to be able to effectively produce software using a given model. In other words, the analysis of the variables determining the values of D and DG (verification of the fulfillment of the condition $D > DG$) allows the determination of what values will enable the use of an arbitrarily chosen approach. The synthesis of the development environment parameters related to this approach may lead to the following question: achieving what values of the TCA and TBA parameters of the project team will allow the use of the *single-form* model approach?

Thus, the proposed condition enables a quantitative assessment of the complexity of the BP process being modeled (parameter D), the effectiveness of the project team (parameter DG), and the selected modeling approach ($D > DG$).

The developed condition was used in a series of experiments, the results of which are presented in the next section.

4. Experimental Results

In order to verify the evaluation method proposed in this paper, both qualitative and quantitative experiments were conducted.

Qualitative experiments are aimed at defining the conditions that should be met by a business process to make its implementation advantageous with the single-form model approach. They were also conducted to assess what determines the choice of method, i.e., which factors (properties of business processes and developers' skills) influence the assessment.

Quantitative experiments are intended to answer the question of which values of important parameters in real business processes and real project teams affect the choice of method the most. On this basis, it can be assessed which method works better for the implementation of real systems.

4.1. Qualitative Experiments

The aim of the qualitative experiments is to assess the variability of the DG, depending on the change in the value of the TCA and TCB project teams' effectiveness coefficients as well as on the change in the size of the VM. In this context, the experiments were divided into three groups.

4.1.1. Assessment of Conditions under Which It Is Advisable to Use the Single-Form Model for the Business Process BP_1

Let us consider a situation where the efficiency of a developer implementing the business process BP_1 from Figure 1 ($m = 15,\ n = 5$) is as follows:

- time to embed one variable field into the form: $t_a = 300$ [s] (5 min),
- time to prepare a blank form: $t_b = 600$ [s] (10 min),
- time to fill one cell of the VM: $t_c = 30$ [s].

The aim of the experiment is to assess for which density values (D) of the VM the use of the single-form model is more advantageous (requires less of the developer's time) than the use of the multi-form model. In practice, a high D density value means a large number of variables shared by business process tasks (the same variables appear in many data entry forms). In the experiment, the value of the variable D was assumed to be successively $D = 0.0666,\ 0.133,\ \ldots,\ 1$. The results of the experiment are presented in Figure 6.

The experiment confirmed the correctness of the conditions developed under Theorem 1. According to the experiment, the single-form model is more favourable when $D > DG = 0.1933$. It is worth noting that in the extreme case of $D = 1$, the implementation of the BP_1 business process using the multi-form model approach is 3.5 times more expensive than in the case of the single-form model.

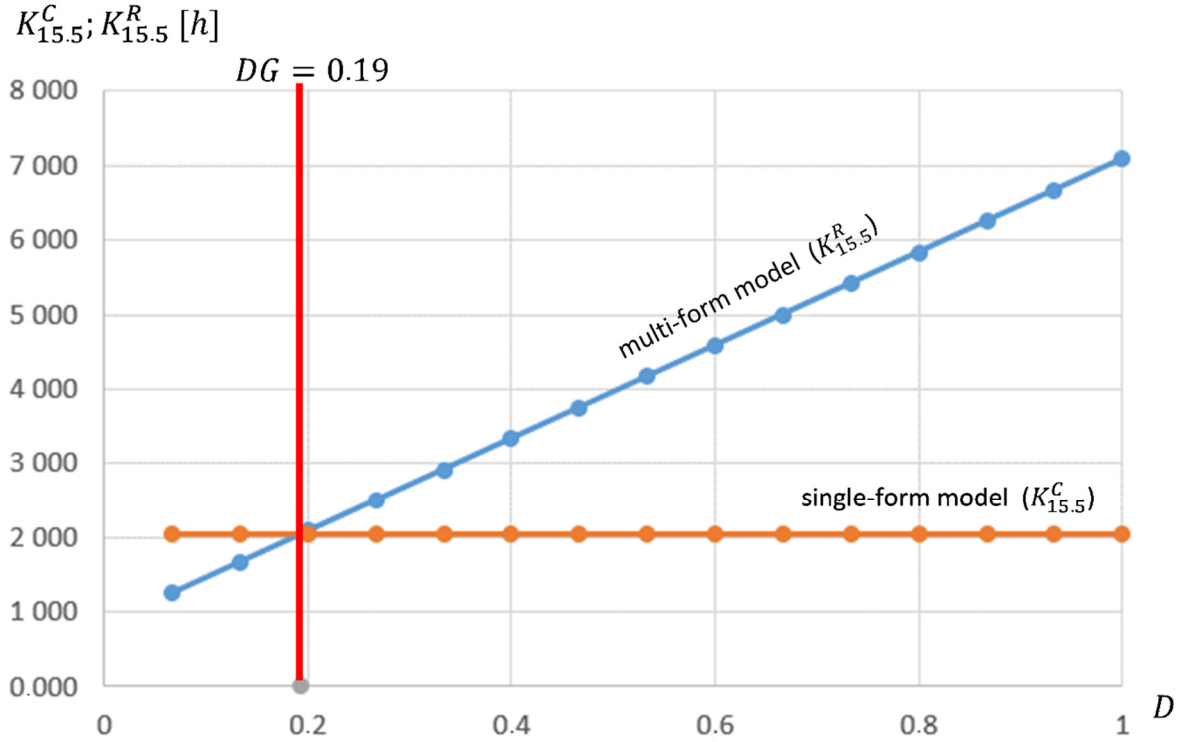


Figure 6. The implementation time of the business process BP_1 depending on the density D.

4.1.2. Assessment of the Variability of the *DG*, Depending on the Change in the Value of the Project Team's Efficiency Coefficients *TCA* and *TCB*

For the purposes of the experiments, it was assumed that the size of the VM is constant and equals $n = 20$; $m = 100$, while the coefficients of effectiveness are variable and take the following values: $TCA \in [0, 1]$; $TBA \in [0, 10]$.

For such parameters, the minimum DG of the VM was determined for which *the single-form model* is less expensive than *the multi-form model*, i.e., $K^C_{m,n} < K^R_{m,n}$.

The obtained results are presented in the diagram in Figure 7. As can be seen, the TCA coefficient has the greatest influence on the value of DG . Along with its increase, DG approaches the value of one, which in practice means that the use of *the single-form model* approach is not profitable. The sensitivity of the DG value to changes in the TBA coefficient is much lower.

The above observation leads to the following conclusion:

A low DG value occurs with a low TCA value. This means that managers should strive to lower the TCA of project teams. In other words, within development teams, it is important to keep the time to fill one cell of the VM (t_c) as short as possible. The smaller it is (in relation to the time to embed one variable field into the form; t_a), the greater the benefit of using the single-form model approach.

4.1.3. Assessment of the Variability of the *DG*, Depending on the Change in the Size of the *VM*

For the purposes of the experiments, it was assumed that the coefficients of efficiency are constant and equal $TCA = 0.033$; $TBA = 4$, while the size of the VM varies in the range $n \in \{1, \ldots, 50\}$; $m \in \{10, \ldots, 500\}$.

For such parameters, the minimum DG of the VM was determined for which the single-form model is less expensive than the multi-form model, i.e., $K^C_{m,n} < K^R_{m,n}$.

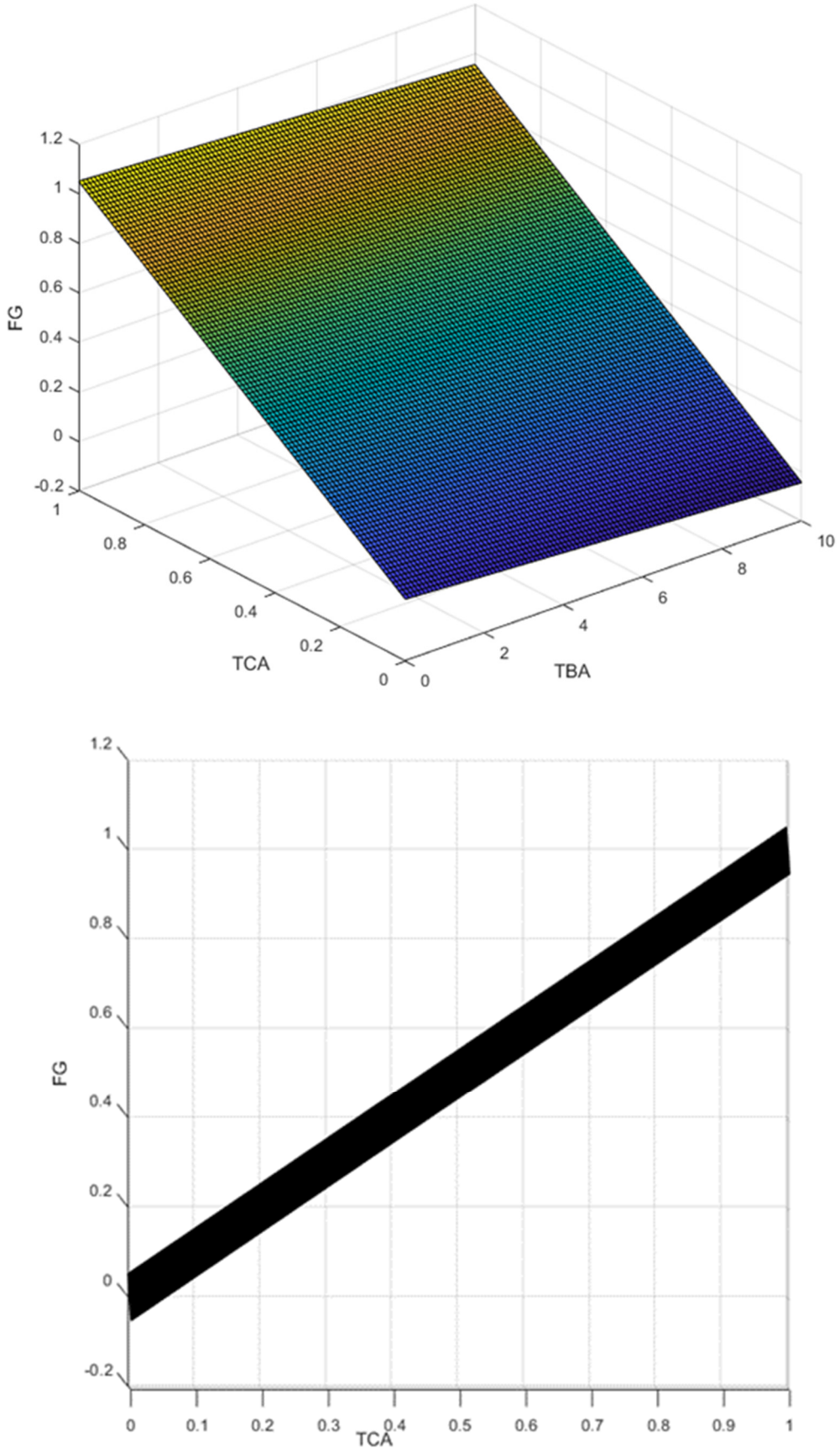


Figure 7. *Cont.*

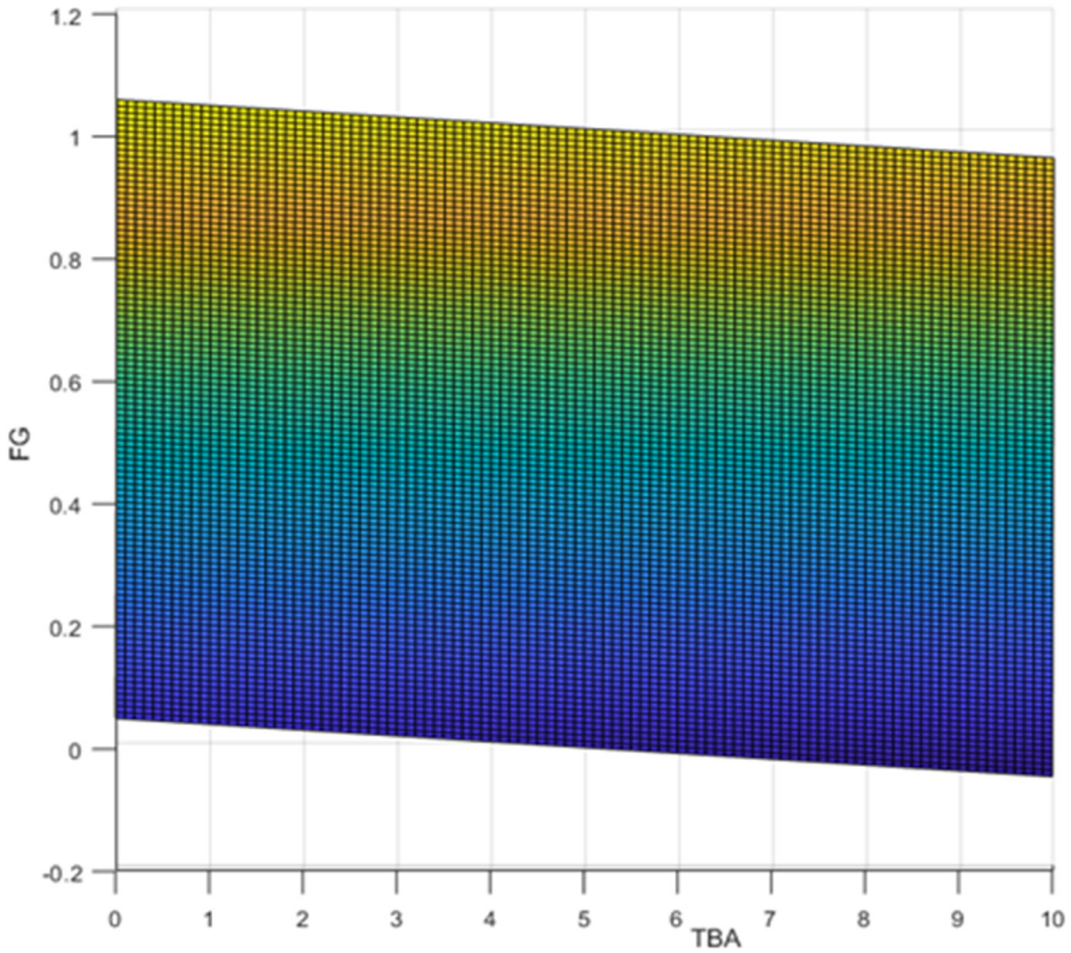


Figure 7. The value of the cost-effective threshold density $DG(TCA, TBA)$ for the matrix of size $n = 20$; $m = 100$. Source: own elaboration.

The obtained results are presented in the diagram in Figure 8. The presented diagrams show that the DG is slightly dependent on the variability of parameters n and m ($DG \in [-0.3; 0.15]$). However, it is worth noting that DG decreases with the increase in the number of tasks and with the decrease in the number of variables.

The single-form model approach is therefore dedicated to processes with a large VM.

4.2. Quantitative Experiments

The purpose of quantitative experiments is to assess the value of the DG and recommend a more favourable modelling approach for real implementations of business processes.

For the purposes of the research, from among nearly 100 business processes, the 30 most-utilised were selected. These processes were implemented in real trade and production enterprises as well as state administration units in 2009–2020. For each of the processes, the following were identified: the number of tasks (n), the number of variables (m), and the visibility matrix density (D). A histogram illustrating the distribution of D values is shown in Figure 9.

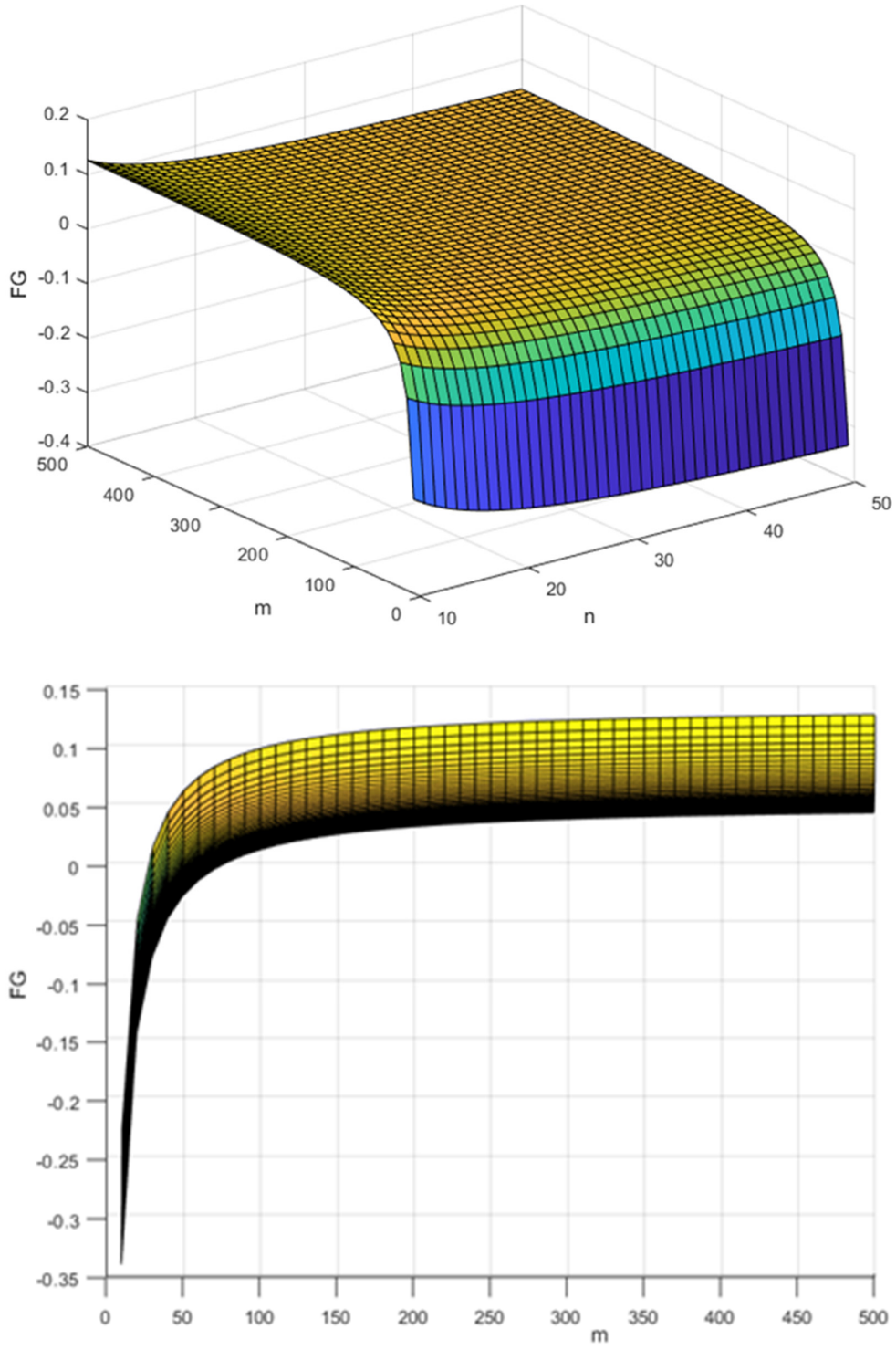


Figure 8. *Cont.*

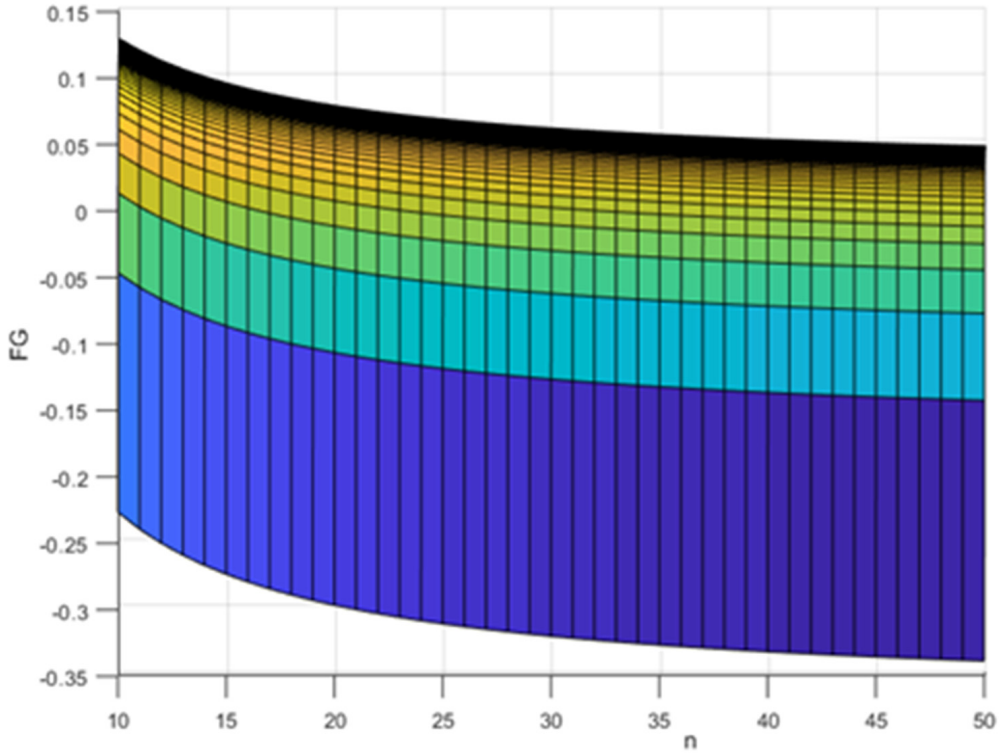


Figure 8. Cost-effective threshold density DG, depending on the change in the size of the VM visibility matrix $DG(n, m)$ for $TCA = 0.033$; $TBA = 4$. Source: own elaboration.

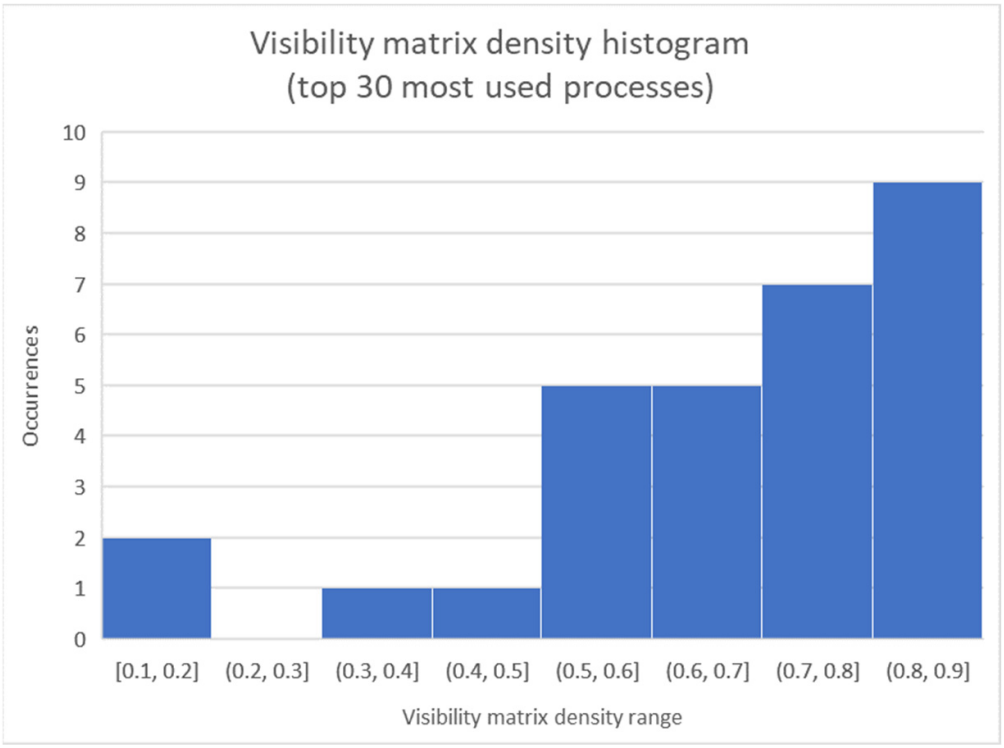


Figure 9. The visibility matrix density histogram for the 30 most-utilised business processes.

As is easy to notice, most of the processes (26 out of 30 analysed processes) are characterised by a density $D \geq 0.52$.

Moreover, the TCA and TBA efficiency coefficients for 11 real developers were identified. The results are presented in Table 4.

Table 4. Efficiency coefficients for 11 real software developers.

Developer	Time to Embed One Variable Field into the Form [s] t_a	Time to Prepare a Blank Form [s] t_b	Time to Fill One Cell of the Visibility Matrix [s] t_c	TCA	TBA
RW	600	900	30	0.05	1.50
AC	900	300	60	0.07	0.33
KW	180	60	20	0.11	0.33
PO	300	900	60	0.20	3.00
JS	300	600	60	0.20	2.00
BK	600	720	30	0.05	1.20
MO	420	420	5	0.01	1.00
JO	60	60	10	0.17	1.00
JM	240	240	20	0.08	1.00
RP	60	180	10	0.17	3.00
AWI	420	60	7	0.02	0.14

The subject of the analysis was the assessment of the DG for each of the analysed business processes and each analysed project team (Table 4). Then, answers were found to the question of whether it is beneficial to use *the single-form model* approach in implementing a given process. For this purpose, the value of density D was determined for each of the processes and compared with the limit value DG. The obtained results are presented in Table 5 and in Figure 10.

Table 5. The density of the visibility matrix for the 30 most-utilised business processes.

Process Id	Task Count n	Variables Count m	Active Variables Count $\sum_{i=1}^{n} f_i$	Density $D = \frac{\sum_{i=1}^{n} f_i}{m \times n}$	Cost-Effective Threshold Density of the Visibility Matrix—DG										Recommended Approach
					RW	AC	KW	PO	JS	BK	MO	JO	JM	RP	AWI	
					TBA = 1.50	**0.33**	**0.33**	**3.00**	**2.00**	**1.20**	**1.00**	**1.00**	**1.00**	**3.00**	**0.14**	
					TCA = 0.05	**0.07**	**0.11**	**0.20**	**0.20**	**0.05**	**0.01**	**0.17**	**0.08**	**0.17**	**0.02**	
51	3	58	97	0.56	0.37	0.40	0.44	0.50	0.51	0.37	0.33	0.49	0.41	0.47	0.35	SM
72	8	208	701	0.42	0.17	0.19	0.23	0.31	0.32	0.17	0.13	0.29	0.20	0.28	0.14	SM
548	21	368	5277	0.68	0.09	0.11	0.16	0.24	0.24	0.09	0.06	0.21	0.13	0.21	0.06	SM
238	7	274	1192	0.62	0.19	0.21	0.25	0.33	0.34	0.19	0.15	0.31	0.22	0.30	0.16	SM
195	1	84	56	0.67	1.05	1.07	1.11	1.20	1.20	1.05	1.01	1.17	1.08	1.17	1.02	MM
457	13	184	1515	0.63	0.12	0.14	0.19	0.26	0.27	0.12	0.08	0.24	0.16	0.23	0.09	SM
140	2	60	102	0.85	0.54	0.56	0.61	0.68	0.68	0.54	0.50	0.66	0.58	0.64	0.52	SM
142	2	40	66	0.83	0.53	0.56	0.61	0.66	0.68	0.54	0.50	0.65	0.57	0.63	0.51	SM
241	7	164	89	0.08	0.19	0.21	0.25	0.33	0.33	0.19	0.15	0.30	0.22	0.29	0.16	MM
753	9	157	628	0.44	0.15	0.18	0.22	0.29	0.30	0.15	0.12	0.27	0.19	0.26	0.13	SM
573	1	106	98	0.92	1.05	1.07	1.11	1.20	1.20	1.05	1.01	1.17	1.08	1.17	1.02	MM
556	6	94	487	0.86	0.20	0.23	0.27	0.34	0.35	0.21	0.17	0.32	0.24	0.31	0.18	SM
576	1	113	90	0.80	1.05	1.07	1.11	1.20	1.20	1.05	1.01	1.17	1.08	1.17	1.02	MM
675	9	193	1082	0.62	0.15	0.18	0.22	0.30	0.30	0.16	0.12	0.27	0.19	0.26	0.13	SM

Table 5. *Cont.*

199	2	81	88	0.54	0.54	0.56	0.61	0.68	0.69	0.54	0.51	0.66	0.58	0.65	0.52	SM/MM
194	2	52	86	0.83	0.54	0.56	0.61	0.67	0.68	0.54	0.50	0.66	0.57	0.64	0.52	SM
141	4	23	84	0.91	0.25	0.31	0.35	0.35	0.38	0.26	0.23	0.38	0.30	0.32	0.26	SM
236	5	53	41	0.15	0.23	0.26	0.31	0.35	0.37	0.23	0.20	0.35	0.27	0.32	0.21	SM/MM
473	2	45	48	0.53	0.53	0.56	0.61	0.67	0.68	0.54	0.50	0.66	0.57	0.63	0.52	SM/MM
342	3	42	84	0.67	0.36	0.39	0.44	0.49	0.50	0.36	0.33	0.48	0.40	0.45	0.35	SM
698	9	194	1093	0.63	0.15	0.18	0.22	0.30	0.30	0.16	0.12	0.27	0.19	0.26	0.13	SM
231	6	126	560	0.74	0.21	0.23	0.28	0.35	0.35	0.21	0.17	0.33	0.24	0.31	0.18	SM
396	11	144	1309	0.83	0.13	0.16	0.20	0.27	0.28	0.13	0.10	0.25	0.17	0.24	0.11	SM
335	2	31	24	0.39	0.53	0.56	0.61	0.65	0.67	0.53	0.50	0.65	0.57	0.62	0.51	SM
589	7	38	187	0.70	0.16	0.20	0.25	0.28	0.30	0.17	0.13	0.29	0.20	0.24	0.16	SM
677	5	147	476	0.65	0.24	0.26	0.31	0.38	0.39	0.24	0.21	0.36	0.28	0.35	0.22	SM
600	5	24	49	0.41	0.20	0.26	0.30	0.30	0.33	0.21	0.18	0.33	0.25	0.27	0.21	SM
745	6	28	125	0.74	0.17	0.22	0.27	0.28	0.31	0.18	0.15	0.30	0.22	0.24	0.18	SM
562	4	27	95	0.88	0.26	0.31	0.35	0.37	0.39	0.27	0.23	0.39	0.31	0.33	0.26	SM
228	10	139	1092	0.79	0.14	0.16	0.21	0.28	0.29	0.14	0.11	0.26	0.18	0.25	0.12	SM

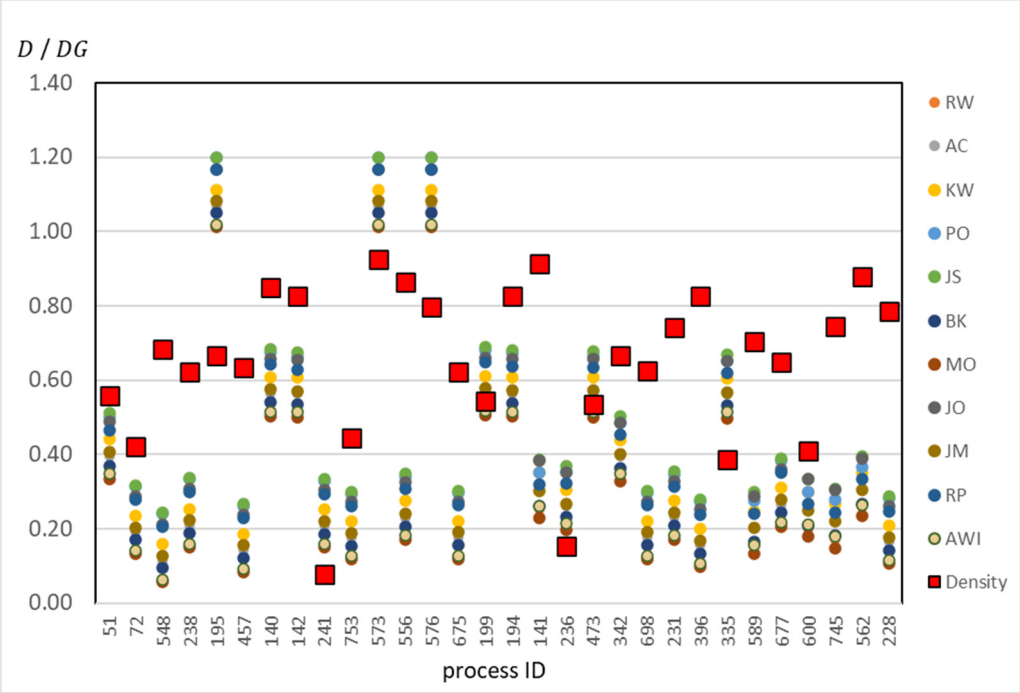


Figure 10. The visibility matrix density (D) and the cost-effective threshold density of the visibility matrix (DG) for business processes from Table 5. Source: own elaboration.

The densities (D) for the business processes under consideration are marked in Figure 10 with the symbol ν, while the DG for subsequent project teams are marked with colours as in the chart legend As is easy to notice, for most of the considered processes it is more advantageous to use the single-form model approach (i.e., the condition $D > DG$ is satisfied). The exceptions are:

- business processes #195, #573, and #576, where the number of tasks equals one ($n = 1$),
- business processes #241 and #236, where the density of the visibility matrix was so low that the use of the single-form model approach is unfavourable ($D < DG$) (in these processes, tasks hardly share variables with each other), and

- business processes #199 and #473, where the advantage of using the single-form model approach depends on the development team ($D \approx DG$). For example, for business process #199, the single-form model approach is recommended for 5 out of 11 developers.

The presented results confirm the conclusions found in the qualitative experiments. Most of the business processes encountered in practice (in this case, 76% of processes) consist of tasks that share most of the variables. In such situations, the value of D of the VM is much higher than the limit value DG resulting from the performance indicators of real development teams (Theorem 1 is satisfied). For such business processes, *the single-form model* approach is recommended.

It is worth emphasising that the higher the D, the greater the benefit of using this approach. To illustrate this relationship, an additional parameter was introduced: $ZK = (K^R_{m,n} - K^C_{m,n})/K^R_{m,n}$. It determines the relative time gain that will be achieved by using the *single-form model*. In other words, the value of ZK corresponds to how much the implementation time of the business process will be shortened (for $ZK > 0$) or extended (for $ZK < 0$) as a result of the use of the single-form model approach. Figure 11 shows the change in the value of ZK depending on the value of D. The calculations were carried out for a process containing $n = 6$ tasks and $m = 82$ variables (the process parameters correspond to the average of the analysed business processes in Table 5). It was assumed that the developers from Table 4 were used to implement the business process. As is easy to see, the value of ZK increases with the D of the VM. After exceeding the value of $D = 0.37$, it is better for all developers to use the *single-form model* approach. It is also worth emphasising that the lower the D, the greater the differences in the value of ZK for different developers. Differences between developers' productivities, however, become blurred with high D values.

The obtained results show that the profit achieved with the use of the single-form model may be as high as 80% (Figure 11). For a business process containing $n = 6$ tasks and $m = 82$ variables, the time gain (for the most productive employee) will be:

$$\mathrm{ZK} = \frac{\left(K^R_{m,n} - K^C_{m,n}\right)}{K^R_{m,n}}$$

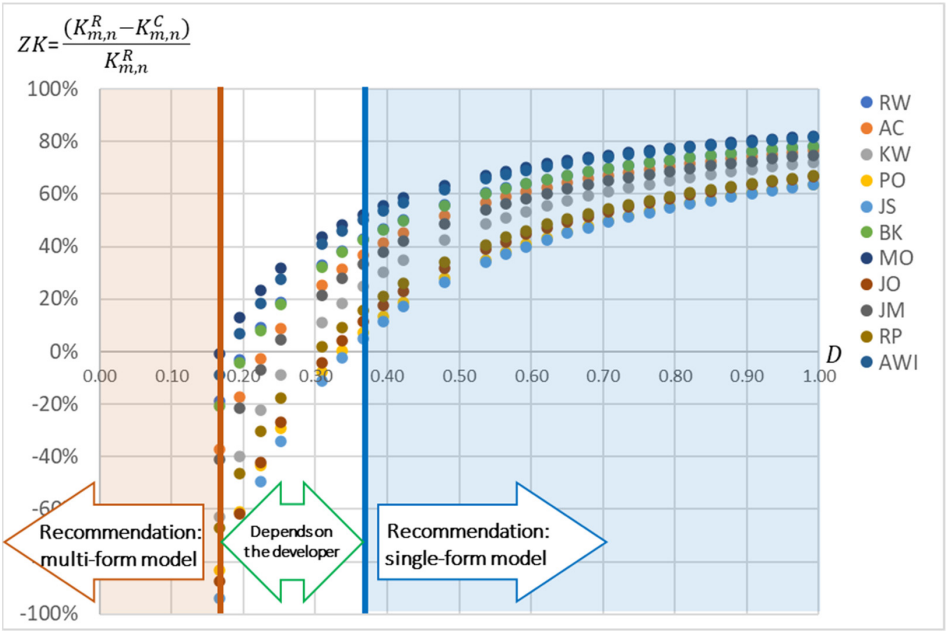


Figure 11. The relative time gain (ZK) depending on the density D for a business process containing $n = 6$ tasks and $m = 82$ variables. Source: own elaboration.

In the extreme case (80%), the use of the single-form model allows for the reduction of the implementation time from 50.73 [h] to 10.37 [h].

5. Conclusions

Tools for building applications based on business process models can use two approaches to build forms: (1) a separate form for each task or (2) one common parameterised form for all tasks. The conducted research answers the question of which approach yields better results from the productivity point of view, i.e., the cost and time of implementing the user interface application.

Qualitative experiments allowed us to determine which properties should characterise the business process so that it would be beneficial to apply the *single-form model* approach to its implementation. They also answered the question of what determines the choice of method, i.e., which factors affect the assessment. It turns out that the greatest impact on productivity in the production of electronic forms that support the tasks of business processes is due to the density of the visibility matrix (D) and the completion times of GUI components, i.e., the project team's productivity (measured by the t indicators meaning, respectively, the time to embed one variable field into the form (t_a), the time to prepare a blank form (t_b), and the time to fill one cell of the VM (t_c)).

Quantitative experiments answered the question of which pragmatic values of the parameters are important for choosing the implementation method for real business processes and real project teams. They allowed us to determine the most common densities of visibility matrices and to determine the actual construction times of data entry forms in specific project teams.

The following conclusions should be emphasised:

1. The choice of the method of building forms to handle tasks in business processes depends on the density of the VM (D) and the productivity coefficients $TCA = t_c/t_a$, $TBA = t_b/t_a$.
2. The most sensitive parameters are TCA and D; the least sensitive parameter is TBA.
3. In practice, the significance of the conclusions from points 1 and 2 above comes down to the following rule: the use of the single-form model is more advantageous when TCA is minimal. This is obtained, for example, when the project manager selects the production technology where setting parameters in the VM requires as little time as possible. This is an important feature of the LCDP that affects productivity.
4. In real IT projects, the densities of the VMs usually exceed the value of $D > 0.69$. That means that most of the process variables are reused among subsequent tasks.
5. In real project teams, the time of building form elements is characterised by a large variance depending on the developer. However, most of the results show significantly shorter parametrisation times for the FM and VM compared to the construction times of separate forms. For most developers, the single-form model approach results in shorter implementation times (only in 2 out of 30 cases was there a situation where the single-form model was unfavourable, and only for some developers).
6. Considering the real productivity of project teams and business process models, in an overwhelming number of cases, it is more beneficial to use the single-form model approach with the use of VM.
7. There are few cases where it is justified to use the multi-form model. These are most often exceptional situations, such as single-task processes and extremely ineffective project teams.

The results of the work presented in this article, in the field of theoretical research, introduces models that allow us to determine the workload needed to build user interfaces as part of supporting business processes on the LCDP. The effort determined in this manner can then be compared for different approaches to user interface implementation. This allows one to determine which approach is most efficient.

From a practical point of view, our work introduces the idea of using a single form to build a user interface that allows one to handle all tasks of the business process. In addition,

they introduce the concept of a VM, which allows one to differentiate the appearance and scope of data presented to the end user when handling tasks.

Our research and the results achieved constitute a benchmark for possible future solutions. This is due to the fact that in the literature so far, the problems of user interface development with different approaches to the data usage modelling in processes have not been discussed. Each subsequent solution can be compared to the multi-form approach and single-form approach described in this article.

The conducted experiments have shown that under certain conditions (Figure 11—green arrow), it is not possible to clearly define which model is always more favourable for a given project team. Therefore, the concept of developing a hybrid model was created. Future work will focus on introducing a hybrid model that will identify subsets of business process tasks for which it will be profitable to use a single form. Such a hybrid approach opens the possibility of even better optimisation of the workload necessary to build user interfaces for the entire business process.

The workload optimisation in this case is understood as the process of assessing and selecting the appropriate modeling approach (analysis process) as well as determining the parameters of the development environment (synthesis process) that guarantee its effectiveness (design time for screen forms). While research in the area of the analysis process is currently encountered in the literature [14,15], the synthesis of the development environment parameters comes down to the application of good practices and gained experience. The methodology of optimizing the development environment parameters (synthesis of the development environment) will enable the use of solutions dedicated directly to companies producing software. These solutions will allow the definition of the team building strategy, principles of cooperation, required performance, etc. The development of this type of methodology is the main goal of future work.

Author Contributions: Conceptualization, R.W.; methodology, R.W. and G.B.; software, R.W.; validation, R.W. and G.B.; formal analysis, G.B. and R.W.; investigation, R.W. and G.B.; resources, R.W.; data curation, R.W.; writing—original draft preparation, R.W.; writing—review and editing, R.W. and G.B.; visualization, G.B. and R.W.; supervision, G.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Sanchis, R.; García-Perales, Ó.; Fraile, F.; Poler, R. Low-Code as Enabler of Digital Transformation in Manufacturing Industry. *Appl. Sci.* **2020**, *10*, 12. [CrossRef]
2. McKendrick, J. *The Rise of the Empowered Citizen Developer*; 2017 Low-Code Adoption Survey; Unisphere Research, a Division of Information Today, Inc.: Medford, NJ, USA, 2017; p. 31.
3. Panayiotou, K.; Tsardoulias, E.; Zolotas, C.; Symeonidis, A.L.; Petrou, L. A Framework for Rapid Robotic Application Development for Citizen Developers. *Software* **2022**, *1*. [CrossRef]
4. Alokla, A.; Gad, W.; Nazih, W.; Aref, M.; Salem, A.-B. Retrieval-Based Transformer Pseudocode Generation. *Mathematics* **2022**, *10*, 604. [CrossRef]
5. Richardson, C.; Rymer, J. New Development Platforms Emerge for Customer-Facing Applications. 2014. Available online: https://www.forrester.com/report/New+Development+Platforms+Emerge+For+CustomerFacing+Applications/-/E-RES113411 (accessed on 19 July 2021).
6. Gartner Inc. Enterprise LCAP (Low-Code Application Platforms) Reviews 2021 | Gartner Peer Insights. Gartner. Available online: https://www.gartner.com/market/enterprise-low-code-application-platform (accessed on 19 July 2021).
7. Alamin, M.A.A.; Malakar, S.; Uddin, G.; Afroz, S.; Haider, T.B.; Iqbal, A. An Empirical Study of Developer Discussions on Low-Code Software Development Challenges. *arXiv* **2021**, arXiv:210311429. Available online: http://arxiv.org/abs/2103.11429 (accessed on 16 July 2021).

8. Bloomberg, J. The Low-Code/No-Code Movement: More Disruptive than You Realize. Forbes. Available online: https://www.forbes.com/sites/jasonbloomberg/2017/07/20/the-low-codeno-code-movement-more-disruptive-than-you-realize/ (accessed on 19 July 2021).
9. Woo, M. The Rise of No/Low Code Software Development—No Experience Needed? *Engineering* **2020**, *6*, 960–961. [CrossRef] [PubMed]
10. Cio, M.F. *What CIOs Need to Know about Low Code Software Development*; CXO Media Inc.: Needham, MA, USA, 2019; p. 4.
11. Bansal, A. 5 Ways a Low Code Digital Automation Platform Can Transform Government Organizations—ProQuest. Available online: https://www-1proquest-1com-1000002id0114.han.wat.edu.pl/docview/2500313137?pq-origsite=primo (accessed on 16 July 2021).
12. Ness, C.; Hansen, M.E. Potential of Low-Code in the Healthcare Sector: An Exploratory Study of the Potential of Low-Code Development in the Healthcare Sector in Norway. 2019. Available online: https://openaccess.nhh.no/nhh-xmlui/handle/11250/2644695 (accessed on 16 July 2021).
13. Alonso, A.N.; Abreu, J.; Nunes, D.; Vieira, A.; Santos, L.; Soares, T.; Pereira, J. Towards a Polyglot Data Access Layer for a Low-Code Application Development Platform. *arXiv* **2020**, arXiv:200413495. Available online: http://arxiv.org/abs/2004.13495 (accessed on 16 July 2021).
14. Wolff, I. *Making In-House Apps with Low-Code, No-Code Platforms*; SME: Southfield, MI, USA, 2019; pp. 59–67.
15. Missikoff, M. A Simple Methodology for Model-Driven Business Innovation and Low Code Implementation. *arXiv* **2020**, arXiv:201011611. Available online: http://arxiv.org/abs/2010.11611 (accessed on 16 July 2021).
16. Turek, P.; Bogacz, P.; Buła, A. Synergia technologii Blockchain i Low-code próbą zwiększenia elastyczności proklienckiej procesów biznesowych przy zachowaniu ich efektywności kosztowej. *Napędy Sterow.* **2020**, *22*, 89–91. Available online: http://yadda.icm.edu.pl/baztech/element/bwmeta1.element.baztech-1c3ef87a-49c3-44b3-aa4f-a94d7c54efe8 (accessed on 16 July 2021).
17. Guerrero, C.V.S.; Lula, B. A Model-Guided and Task-Based Approach to User Interface Design Centered in a Unified Interaction and Architectural Model. In *Computer-Aided Design of User Interfaces III*; Kolski, C., Vanderdonckt, J., Eds.; Springer: Dordrecht, The Netherlands, 2002; pp. 119–129. [CrossRef]
18. Akiki, P.A.; Bandara, A.K.; Yu, Y. Adaptive Model-Driven User Interface Development Systems. *ACM Comput. Surv.* **2014**, *47*, 9:1–9:33. [CrossRef]
19. Ammar, L.B. An Automated Model-Based Approach for Developing Mobile User Interfaces. *IEEE Access* **2021**, *9*, 51573–51581. [CrossRef]
20. Milosavljević, G.; Ivanović, D.; Surla, D.; Milosavljević, B. Automated construction of the user interface for a CERIF-compliant research management system. *Electron. Libr.* **2011**, *29*, 565–588. [CrossRef]
21. Heimann, D.I. CATS-an automated user interface for software development and testing. In Proceedings of the 1996 Annual Reliability and Maintainability Symposium, Las Vegas, NV, USA, 22–25 January 1996; pp. 163–166. [CrossRef]
22. Lieberman, H. Computer-Aided Design of User Interfaces by Example. In *Computer-Aided Design of User Interfaces III*; Kolski, C., Vanderdonckt, J., Eds.; Springer: Dordrecht, The Netherlands, 2002; pp. 1–12. [CrossRef]
23. Kolski, C.; Vanderdonckt, J. (Eds.) *Computer-Aided Design of User Interfaces III. Proceedings of the Fourth International Conference on Computer-Aided Design of User Interfaces, Valenciennes, France, 15–17 May 2002*; Springer: Dordrecht, The Netherlands, 2002. [CrossRef]
24. Brandl, A. Concepts for Generating Multi-User Interfaces Including Graphical Editors. In *Computer-Aided Design of User Interfaces III*; Kolski, C., Vanderdonckt, J., Eds.; Springer: Dordrecht, The Netherlands, 2002; pp. 167–178. [CrossRef]
25. Kaindl, H.; Jezek, R. From Usage Scenarios to User Interface Elements in a Few Steps. In *Computer-Aided Design of User Interfaces III*; Kolski, C., Vanderdonckt, J., Eds.; Springer: Dordrecht, The Netherlands, 2002; pp. 91–102. [CrossRef]
26. Paternò, F.; Santoro, C. One Model, Many Interfaces. In *Computer-Aided Design of User Interfaces III*; Kolski, C., Vanderdonckt, J., Eds.; Springer: Dordrecht, The Netherlands, 2002; pp. 143–154. [CrossRef]
27. Trætteberg, H. Using User Interface Models in Design. In *Computer-Aided Design of User Interfaces III*; Kolski, C., Vanderdonckt, J., Eds.; Springer: Dordrecht, The Netherlands, 2002; pp. 131–142. [CrossRef]
28. van Hemert, J.; Koetsier, J.; Torterolo, L.; Porro, I.; Melato, M.; Barbera, R. Generating web-based user interfaces for computational science. *Concurr. Comput. Pract. Exp.* **2011**, *23*, 256–268. [CrossRef]
29. Bocciarelli, P.; D'Ambrogio, A.; Panetti, T.; Giglio, A. E-MDAV: A Framework for Developing Data-Intensive Web Applications. *Informatics* **2022**, *9*, 12. [CrossRef]
30. Cummings, M.A. The Development of User Interface Tools for the Computer Aided Prototyping System. Master's Thesis, Defense Technical Information Center, Fort Belvoir, VA, USA, 1990; p. 345.
31. Engel, J.; Märtin, C.; Herdin, C.; Forbrig, P. Formal Pattern Specifications to Facilitate Semi-automated User Interface Generation. In Proceedings of the Human-Computer Interaction: Human-Centred Design Approaches, Methods, Tools and Environments, Las Vegas, NV, USA, 21–26 July 2013; pp. 300–309. [CrossRef]
32. Dodig-Crnkovic, G.; Ljungblad, S.; Obaid, M. 4th Space as Smart Information Ecology with Design Requirements of Sustainability, Ethics and Inclusion. *Proceedings* **2022**, *81*, 1124. [CrossRef]
33. Engel, J.; Herdin, C.; Märtin, C. Evaluation of Model-Based User Interface Development Approaches. In Proceedings of the Human-Computer Interaction, Theories, Methods, and Tools, Heraklion, Greece, 22–27 June 2014; pp. 295–307. [CrossRef]

34. Gribova, V. Methods for Decreasing Time and Effort during Development and Maintenance of Intellectual Software User Interfaces. In Proceedings of the Advanced Intelligent Computing Theories and Applications. With Aspects of Theoretical and Methodological Issues, Shanghai, China, 15–18 September 2008; pp. 792–799. [CrossRef]
35. Bolcer, G.A. User interface design assistance for large-scale software development. *Autom. Softw. Eng.* **1995**, *2*, 203–217. [CrossRef]
36. Cruz, E.; Machado, R.-J.; Santos, M. From Business Process Modeling to Data Model: A Systematic Approach. In Proceedings of the 2012 Eighth International Conference on the Quality of Information and Communications Technology, Lisbon, Portugal, 3–6 September 2012; p. 210. [CrossRef]
37. Meyer, A.; Smirnov, S.; Weske, M. *Data in Business Processes*; Universität Potsdam: Potsdam, Germany, 2011.