computers

MDPI

Article

A Lightweight BPMN Extension for Business Process-Oriented Requirements Engineering

Benedetto Intrigila [1], Giuseppe Della Penna [2,*] and Andrea D'Ambrogio [1]

[1] Department of Enterprise Engineering, University of Rome Tor Vergata, 00133 Rome, Italy; benedetto.intrigila@uniroma2.it (B.I.); dambro@uniroma2.it (A.D.)
[2] Department of Information Engineering, Computer Science and Mathematics, University of L'Aquila, 67100 L'Aquila, Italy
* Correspondence: giuseppe.dellapenna@univaq.it

Abstract: Process-oriented requirements engineering approaches are often required to deal with the effective adaptation of existing processes in order to easily introduce new or updated requirements. Such approaches are based on the adoption of widely used notations, such as the one introduced by the Business Process Model and Notation (BPMN) standard. However, BPMN models do not convey enough information on the involved entities and how they interact with process activities, thus leading to ambiguities, as well as to incomplete and inconsistent requirements definitions. This paper proposes an approach that allows stakeholders and software analysts to easily merge and integrate behavioral and data properties in a BPMN model, so as to fully exploit the potential of BPMN without incurring into the aforementioned limitation. The proposed approach introduces a lightweight BPMN extension that specifically addresses the annotation of data properties in terms of constraints, i.e., pre- and post-conditions that the different process activities must satisfy. The visual representation of the annotated model conveys all the information required both by stakeholders, to understand and validate requirements, and by software analysts and developers, to easily map these updates to the corresponding software implementation. The presented approach is illustrated by use of two running examples, which have also been used to carry out a preliminary validation activity.

Keywords: requirements engineering; requirements elicitation; business process management; BPMN

Citation: Intrigila, B.; Della Penna, G.; D'Ambrogio, A. A Lightweight BPMN Extension for Business Process-Oriented Requirements Engineering. *Computers* **2021**, *10*, 171. https://doi.org/10.3390/computers10120171

Academic Editor: Dae-Kyoo Kim

Received: 9 November 2021
Accepted: 14 December 2021
Published: 16 December 2021

1. Introduction

Requirements engineering is the essential software engineering process for eliciting, specifying, verifying and maintaining the requirements of a software product throughout its lifecycle [1]. As stressed by several authors (see, e.g., [2,3]), requirements analysis is the most critical phase in the software development process, due to the large impact on costs of fixing requirements faults. Among the various issues that may lead to the introduction of faulty requirements in terms of, e.g., ambiguity, incompleteness and inconsistency, the heterogeneity of the different stakeholders and their relevant views is particularly significant.

This paper addresses such an issue by focusing on the extension of a business process notation, namely, BPMN (Business Process Management and Notation) [4], which has proven to be easy to understand for the various stakeholders involved in a software requirements engineering effort, minimizing the risks of misunderstandings and omissions (on the omission problem see, e.g., [5]).

The adoption of business process driven approaches to carry out requirements engineering efforts has been widely investigated by several authors, as well as applied to various application domains [6,7].

The use of notations that provide an effective support to both business process management and requirements engineering approaches allows business and software analysts to bridge the gap between them and thus take advantage of both approaches [8].

BPMN is now considered as the de facto standard in the business process management field. However, when applied to the definition and specification of software requirements, it shows significant limitations in terms of data modeling [9]. This is mainly due to the nature of BPMN, which has been introduced to provide a clear, concise and flexible language for the specification and analysis of business processes, thus focusing on process modeling rather than on data modeling.

The data or entities that flow among the various activities of a given process are only defined in an abstract way in a BPMN model. In order to overcome such a limitation, this paper introduces a *lightweight* extension, which specifically addresses the annotation of the structural properties that are essential to provide a complete specification of software requirements. The extension is denoted as lightweight because it does not alter the structure of a BPMN model. As such, an extended model is still a valid BPMN model.

At the same time, the proposed extension is flexible enough to have direct control over what to add in terms of data modeling (using *ad hoc annotations*), in order to enable an effective requirements definition, which aims to be easy to understand for both stakeholders and software developers in charge of implementing or updating the underlying information system.

As a result, BPMN models can be effectively applied to provide an integrated specification of both behavioral and data properties, by limiting the annotations of data properties to those entities that are of interest to fully catch the stakeholders' needs and make them clearly understandable for software developers.

Indeed, data properties are annotated as constraints, i.e., the pre- and post-conditions that the different activities of a business process must satisfy. These constraints are, in turn, built upon a simplified view of the underlying information system, composed by *entities* with *attributes* that are meaningful for the specific requirement and its related process activities. Moreover, key attributes involved in the requirement may be highlighted by associating a *visual hint*, such as a specific colour, which is directly edited on the BPMN diagram.

The resulting annotated BPMN model provides the following advantages:

1. Stakeholders can define and use entities and attributes at the desired level of abstraction, without dealing with the actual implementation details;
2. The entities and their attributes are defined on demand, i.e., when requirements refer to data properties relevant to the BPMN branching (i.e., when an edge is guarded by a certain data property or expression), without introducing unnecessary information;
3. The BPMN model clearly specifies where attribute values are used and how they are changed by the different activities, so to enhance model understandability.

To sum up, the result is a BPMN-centric requirements elicitation and definition process, where the visual representation of the annotated model conveys all the information required both by stakeholders, to understand and validate requirements, and by software analysts and developers, to easily map these updates to the corresponding software implementation.

The proposed approach is not intended to replace existing requirements engineering methodologies, e.g., object-oriented analysis methods [10] or agile methodologies [11]. It is rather intended to support business process management efforts that lead to new or updated requirements for the underlying information system implementation.

This paper's proposal is particularly effective in application domains that are characterized by a significant variability in terms of stakeholder needs and constraints [12], such as in the healthcare domain, where business processes may require frequent updates in order to address new recommendations given by public authorities (see, e.g., [13]). In these domains, the use of requirements engineering based on business processes requires software analysts to quickly react to process changes that have an impact in terms of new or updated requirements.

The paper is organized as follows. Section 2 summarizes the related work about requirements engineering exploiting business process models. Section 3 briefly describes the requirements analysis approach introduced in this paper. Section 4 exemplifies such approach on a healthcare-related case study, comparing its outcome with that of a conven-

tional requirement integration process. Then, in Section 5, the proposed contribution is applied to another real-world case study, in a different application field, in order to better evaluate its generality, applicability and limitations, which are then discussed in detail in Section 6. Finally, Section 7 shows the prototypal tool developed to support the presented approach, and Section 8 outlines the conclusions.

2. Related Work

Business process management (BPM) approaches, which have contributed to bridging the gap between business and IT sectors in organizations, are often associated with the adoption of information systems that implement the designed business processes, so to support process improvement efforts. Typically, these information systems are required to quickly adapt to business process updates in a frequently changing business scenario [12]. As such, business processes are to be taken into account during the requirements engineering phase [8].

Business process is defined by the use of notations that must be understandable and useful to business analysts. As aforementioned, BPMN is currently used as the reference notation for the definition of business process and for the associated software requirements engineering activities [9].

The use of BPMN-based approaches to support requirements engineering and information systems modeling has been investigated by several authors, mostly covering the development of methodologies, languages and heuristics to derive software requirements from business process models, according to various requirements engineering methodologies, as summarized in [14].

The need to cope with the limitations of BPMN in terms of data modeling is seen as a common issue to be dealt with by the introduction of proper BPMN extensions. The contribution in [15] effectively summarizes and reviews all such extensions, which are provided either by using the native extension mechanisms of BPMN (see, e.g., [16]) or by formally applying metamodel-based approaches introduced in the field of model-driven engineering to specify domain-specific modeling languages (see, e.g., [17,18]).

Similarly in terms of data modeling, which is dealt with either by introducing informal (natural-language-based) annotations to define data requirements at a significantly high level of abstraction or by using more formal approaches that allow a detailed specification of domain-specific data structures manipulated by process activities [15].

The contribution proposed in this paper differs from all such previous contributions in terms of the *extension mechanism*, which is based on an informal yet precise annotation of BPMN models, that does not alter the model structure, and *versatility*, being applicable to various application domains (as shown by the case studies described in the next sections).

As aforementioned, the proposed contribution is not intended to be applied to specific requirements engineering methodologies. However, the BPMN-based extension introduced in this paper provides an effective support for managing requirements integration tasks through a simple iterative approach, which can be effectively applied to *agile methodologies* [8]. Indeed, whilst agile methodologies want to avoid complex documentation, they actually need an effective requirements elicitation and lean documentation. As an example, the proposed approach could be applied to a SCRUM methodology, in which BPMN models could be used to convey the stakeholders' needs, so to properly build requirements documentation in terms of so-called user stories [19]. In such a case, the proposed approach could be used either by preserving the original SCRUM process, thus making BPMN models not part of the *product backlog* (the list of maintained artifacts), or by adding BPMN models to the product backlog, in order to properly integrate user stories, as reported in [20].

3. The Requirement Analysis Process

Requirement analysis, assisted by the proposed lightweight BPMN extension, is accomplished through an iterative review process, whose overall structure can be illustrated

as in Figure 1. The process has assigned roles and, in particular, a unique owner, which is typically a manager, and several stakeholders, which are involved in determining the organizational solution.

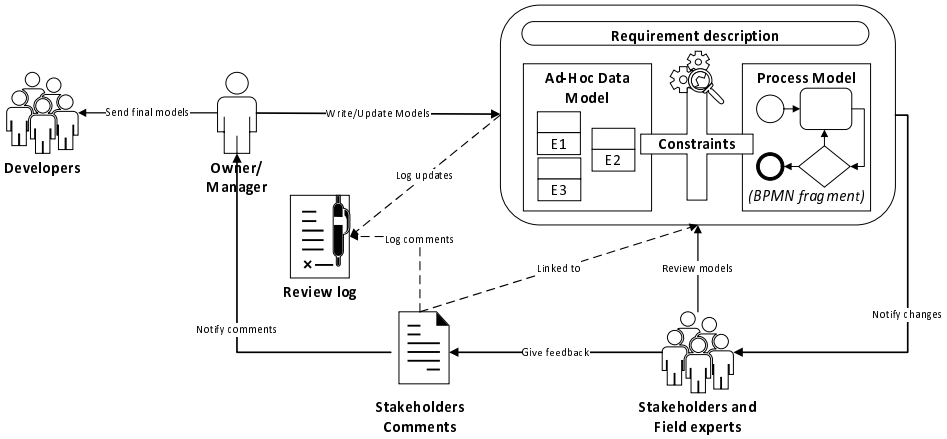

Figure 1. The overall requirement analysis process.

As shown in Figure 1, the process is started by the owner/manager who first writes a natural language description of the new or updated requirement and defines the process model to modify in order to implement it. Here, the process model is a BPMN fragment commonly extracted from the overall BPMN model of the system under review.

Then, the manager defines the ad hoc data model to be used in the review process. The design of such data model is at the heart of the proposed methodology: indeed, in general, the high level procedural description of requirements often needs to reference details related to the low level structure of the underlying information system. However, such details, which may be useful for the IT staff, are hard to understand for most stakeholders, which should instead focus on the correct integration of the requirements into the high level process. To mask this (unnecessary) complexity to the requirements analysis process, we create such a simplified data model, which is then used in the BPMN model. To this aim, we adopt the following rules, so to obtain the desired simplification while maintaining traceability to the underlying implementation:

1. For each attribute and entity cited, a description must be given, which clarifies its intended meaning, so that the IT staff can understand how it can be derived from the real data model of the information system;
2. Complex expressions can (and should) be synthesized in a single attribute of a suitable entity. In the definition of such *complex attributes*, one should choose a bottom-up approach and build them upon other attributes with lower complexity. This would not impact the BPMN representation, but it could give some high-level hints to the developers about how the attribute should be actually derived;
3. By properly building complex attributes, one should try to isolate a small number of *key attributes*, which we shall call *dominant*, that are at the core of the requirement. These dominant attributes will be given a specific visual representation in the BPMN.

The actual meaning of these rules will be clarified in the case studies described in the following sections.

Once the data model is defined, the manager has an "as is" (fragment of) BPMN to work on, as well as a set of attributes that should guide its update. He may now add to the BPMN appropriate new activities and write pre-/post-conditions, based on the above data model attributes, in order to accomplish the new requirement. In general, this is accomplished by placing appropriate comments on the ingoing (for preconditions) and outgoing (for postconditions) arrows of the activities. As a further expedient to simplify the process analysis, we give special treatment to the *dominant attribute*: indeed, we do not place its values on the diagram, but rather we map the possible values to a small number

of meaningful subsets and associate each subset with a visual hint, i.e., a colour, which will be actually placed in the BPMN comments. A suitable definition of the domain data model should allow users to place very simple expressions in these comments, such as single values or ranges. If more complex expressions (e.g., formulas) are needed, one should return to the previous step and try to introduce further (complex) attributes to make such expressions simpler.

All the involved stakeholders review the BPMN model (and the corresponding data model) and give their feedback. Note that here we assume that the stakeholders are in a subordinate position with respect to the owner, so they cannot modify the BPMN model; rather they can suggest to the owner the wanted modifications, but it is of course possible to create a more cooperative environment where every stakeholder can be enabled to directly modify such document.

The manager uses the feedback to further refine the BPMN model and/or the data model, and the process iterates until the manager believes that the stakeholders' comments do not require further changes.

Finally, the review artefacts can be sent to the developers in order to update the software accordingly.

4. Requirement Analysis: The Hip Fracture Treatment Case Study

In order to illustrate our methodology, in this section, we introduce a real-world case study. In particular, we start by describing a set of requirements that were integrated in a pre-existing healthcare process related to the *treatment of a hip fracture*. Originally, such integration followed an ad hoc approach, supported by several meetings, which produced informal specification documents that were later implemented in the hospital information system as well as in the process manuals distributed to the hospital staff. Some of the staff members involved in the original process have been asked to repeat the integration—this time following the new methodology proposed in this paper. At the end of the process, they were asked to report their feedback, comparing the effort spent to integrate the requirement through the new process with their previous experience.

The new requirements are the following:

1. The entire procedure must be completed within 48 h;
2. If the age of the patient is greater than 60 years, the patient must be subjected to the osteoporosis test before the surgery takes place;
3. The patient must have a sufficient pain relief before the surgery takes place;
4. When the patient is transferred from the emergency room to the hospital (orthopaedic) ward, a nurse or a physician must be present in the ward, who speaks a language spoken by the patient.

We recall that the proposed approach assumes that a BPMN model of the existing process already exists. Therefore, we first asked the staff to determine the fragment of the current BPMN that would be actually affected by these new requirements. The result is shown in Figure 2.

Since Requirement 1 involves a temporal constraint, it can be directly expressed in the BPMN model using timed activities and timers. On the other hand, Requirements 2–4 have a more complex structure, which includes several conditions. Therefore, in the following, we will focus on the analysis of such requirements.

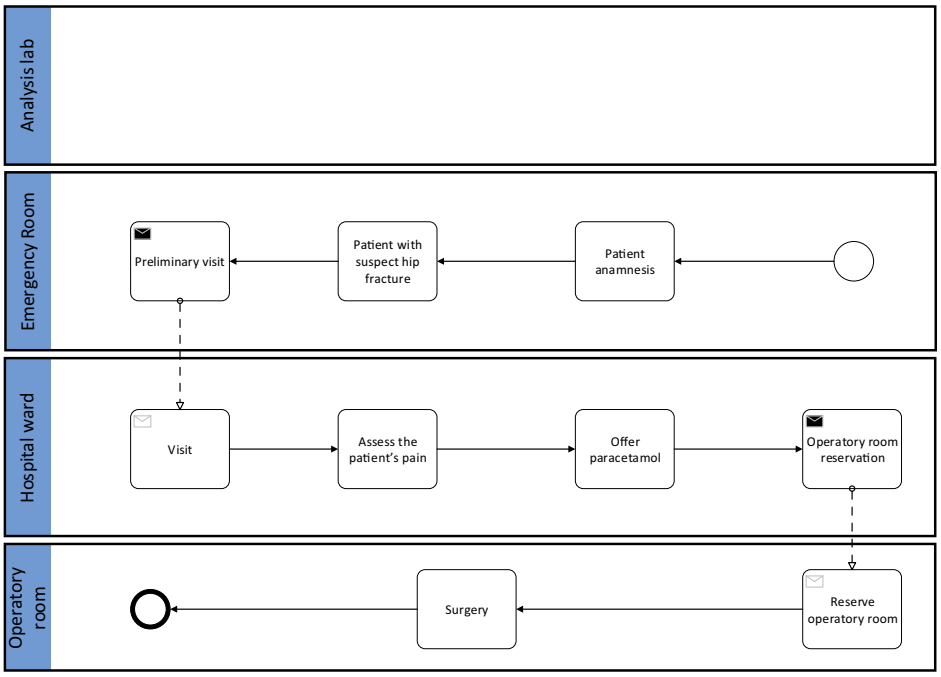

Figure 2. Hip fracture treatment on patients with more than 60 years: initial BPMN.

4.1. Analysis of Requirement 2

4.1.1. Requirement Processing

Requirement 2 describes a *constrained action*: we can indeed split it as follows

- The *action* to perform "perform osteoporosis test before surgery";
- The *preconditions*, i.e., the conditions that must hold for the action to be applicable: "the patient must have a suspected hip fracture" and "the patient must be over 60 year old".

We may note that, given the assumptions of our case study, the first precondition always holds. Moreover, the action is clearly unspecified: what is the relation between the test and the surgery? Actually, the test is needed since surgery cannot be performed if the patient suffers from osteoporosis (positive test). Thus, our initial action hides another set of preconditions that actually guard the "surgery" action. The final, expanded requirement structure can be then formulated as follows:

"If (*precondition*) the patient has a suspected hip fracture and (*precondition*) the patient is over 60 year old then (*action*) perform osteoporosis test. When (*precondition*) the results are available, if (*precondition*) the osteoporosis test gives negative result then (*action*) perform surgery".

Note that, for sake of simplicity, here we will leave unspecified the case where the test is positive and mark this issue on the BPMN with an *exception*.

4.1.2. Domain Data Model Building

The preconditions extracted from Requirement 2 refer to attributes of specific entities involved in the process. In particular, the requirement refers to the age of the patient, i.e., to the attribute "*age*" of the "*patient*" entity. While it is correct to assume that the "*patient*" entity is already defined in the hospital information system, in general, such kind of entity has a complex structure, consisting of many details that are not relevant for the current requirement. Moreover, the "*age*" attribute may not be present, since it is clearly derivable from the date of birth. Finally, the requirement makes reference to another value, namely, the osteoporosis test results. In this case, we are not addressing an attribute of a specific entity: to obtain such a value from the hospital information system, we probably need to perform several queries on different entities, combine the results and give them an interpretation (i.e., comparing values with admissible ranges).

Therefore, following the methodology described in Section 3, we create an *ad hoc domain data model* consisting of a *patient* entity, containing two attributes: a numerical *age* attribute and an *osteoporosis_test* enumerated attribute, which can assume the values "not_required", "required", "to_assess", "positive", and "negative", and will be our *dominant* attribute. In particular, we will visually represent such dominant attribute using white for "required", grey (with the common meaning of "uncertain") for "to_assess", green ("ok, proceed") for "not_required" and "negative", and red ("stop, exception") for "positive", as shown in Figure 3.

Entity	Attribute	Description	Value	Color
patient	age	the age of the patient	*0-100*	
	osteoporosis_test	*not_required* if the age is less than or equal to 60, *required* if age is more than 60 and the test has not been performed, *to_assess* while waiting for the test results, *positive* or *negative* after the interpretation of the test results.	*not_required*	green
			negative	
			required	white
			positive	red
			to_assess	grey

Figure 3. Data model for the "osteoporosis" requirement (2) in the hip fracture treatment.

The rationale behind this choice is that we need the age to first choose if the test must be performed or not, but then, in the rest of the process, we need only the dominant *osteoporosis_test* attribute, which is built also upon the age (i.e., it is set to "not_required" if the age is less than 60 years old), to determine the BPMN path to follow.

The full description of the attributes and their values is given in Figure 3.

4.1.3. BPMN Mapping

As the next step, we need to add to the BPMN the appropriate new activities, i.e., *Osteoporosis analysis request* and *Read results*, and write their pre- and/or post-conditions, based on the attributes above.

The resulting BPMN is shown in Figure 4 where, for sake of clarity, we omitted some activities (included in the initial BPMN of Figure 2) that are not relevant for the current requirement analysis. In the figure, the *age* attribute is used as a first condition and, once checked, continues to be implicitly considered in the higher-level *osteoporosis_test* attribute, which is represented by the coloured squares that act as preconditions for the "operatory room reservation" and "osteoporosis analysis request" activities and postconditions for the "osteoporosis analysis" activity.

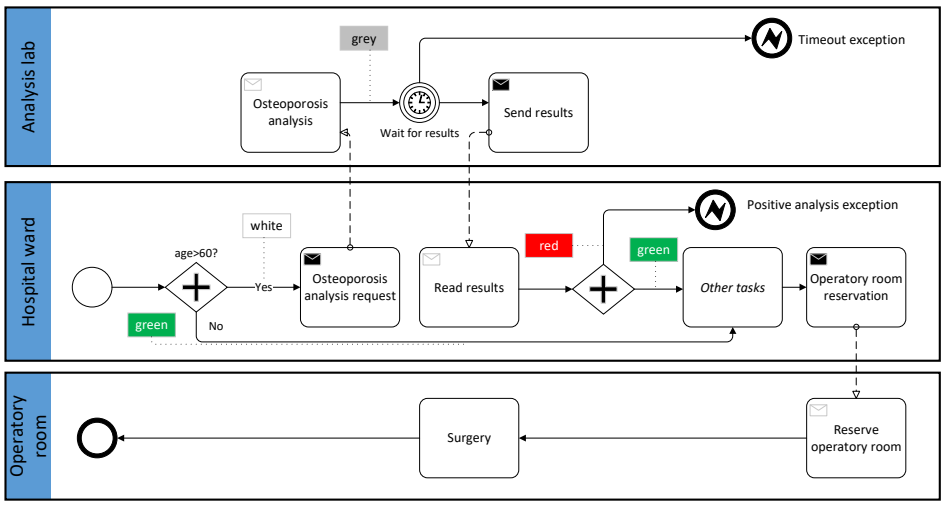

Figure 4. Hip fracture treatment: final BPMN for requirement (2).

Compared to the actual implementation of the requirement "manually" derived by the hospital staff, the solution obtained through our methodology was exactly the same. Indeed, the requirement and its fulfillment were so clear and straightforward in this

case that the application of our methodology could not help in finding any ambiguity or error. However, the formal documentation created as the result of the proposed process can be seen as a collateral advantage, since it may be used as a solid basis for further integration steps.

4.2. Analysis of Requirement 3

Requirement 3 says that the patient must have a sufficient pain relief before being operated. Again, starting from the BPMN in Figure 2, this can be easily seen as a precondition constraint on the "operatory room reservation" activity involving a *pain* attribute of the *patient* entity, which is dominant for this requirement and has three possible values: "to_assess", "no_pain" and "pain", as shown in Figure 5.

Entity	Attribute	Description	Value	Color
patient	pain	*pain* if the user complains of a pain	*to_assess*	grey
			no_pain	green
			pain	red

Figure 5. Data model for the "pain relief" requirement (3) in the hip fracture treatment.

Initially the pain is still *to_assess*. Then, the required action is to offer different analgesics until the pain precondition evaluates to *no_pain*.

If we encode the values of our dominant attribute using the grey, green and red colors, respectively, we obtain the modified BPMN depicted in Figure 6.

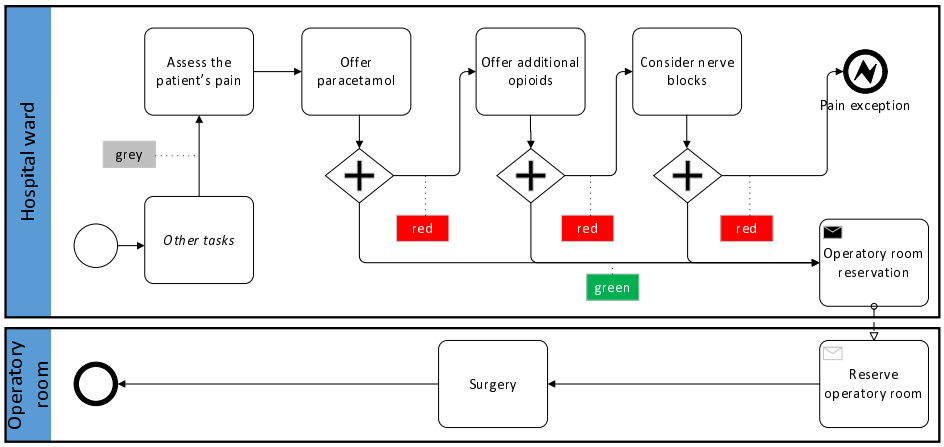

Figure 6. Hip fracture treatment: final BPMN for requirement (3).

Additionally, in this case, the application of our methodology led to no particular benefit to the integration itself, given the simplicity of the requirement, but helped in producing a suitable documentation of the overall updated process that could be useful to both produce the corresponding unambiguous human-readable documentation and, especially, to instruct the IT staff on the implementation changes required to fulfill the new requirement.

4.3. Analysis of Requirement 4

Finally, the last requirement "a patient can be transferred to a specific hospital ward only if there is a staff member that speaks his/her language" presents a more complex implementation.

Here, we need to include three entities in our analysis: *physician*, *nurse* (composing the hospital staff) and *patient*. Actually, following the rules described in the previous sections, we could simply define a dominant attribute *has_common_language_with_staff* and place it on the patient entity.

In this case, however, we are facing two problems: first, obtaining the value of the attribute above, i.e., determining if the patient and the staff share a common language,

is a complex task that can be achieved in many ways. Second, the "spoken languages" information is not guaranteed to be present in the hospital information system. Therefore, we need to carefully define our attributes in order to suggest the IT staff that this information should be added to the real data model of the system and then used to derive the *has_common_language_with_staff* value by a simple set operation. The resulting attribute definition is given in Figure 7.

Entity	Attribute	Description	Value	Color
physician	languages	languages spoken by the physician	*List of languages*	
nurse	languages	languages spoken by the nurse	*List of languages*	
patient	languages	languages spoken by the patient	*List of languages*	
	has_common_language_with_staff	*true* if the patient languages share a common value with the ones of physicians or nurses	*to_assess*	grey
			yes	green
			no	red
			not_at_moment	yellow

Figure 7. Data model for the "common language" requirement (4) in the hip fracture treatment.

We can then define the four possible values for our dominant attribute *has_common_language_with_staff*, that are "to_assess", "yes", "no" and "not_at_moment" (i.e., a staff member exists who speaks a common language with the patient, but he/she is not currently available), which we map to the grey, green, red and yellow colours, respectively. Figure 8 shows how these colours are placed on the BPMN.

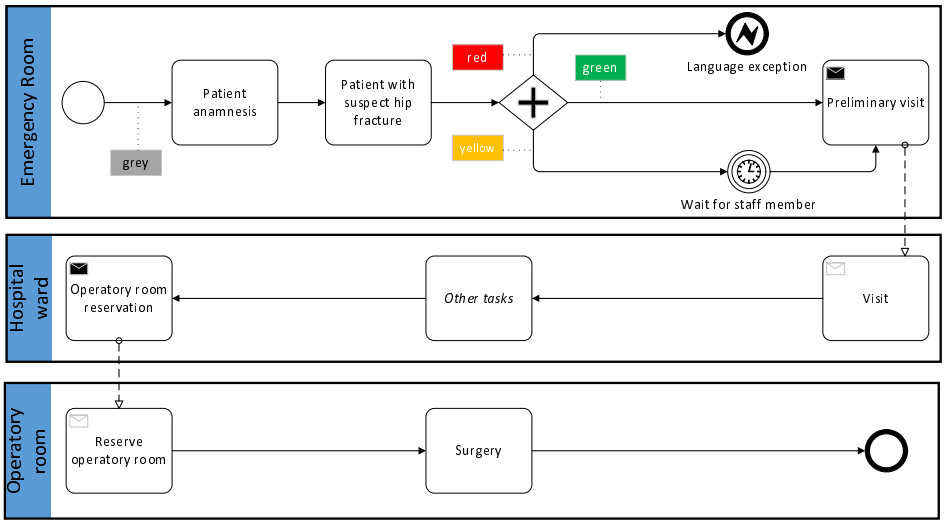

Figure 8. Hip fracture treatment: final BPMN for requirement (4).

Note that the patient enters in the hospital emergency room with a common language *to_assess* (precondition) and exits with a *yes* or *no* value (postcondition). This also implicitly tells that the place to verify this new requirement is the emergency room.

In this case, the application of our methodology quickly converged to a correct solution for the integration of the requirement, whereas the "manual" approach actually employed by the hospital staff initially led to a restrictive solution, which proved to be faulty in many cases. Indeed, the initial solution exploited the *patient nationality*, deducible from his documents, to look for physicians/nurses who could communicate with him. However, the nationality does not directly imply the languages spoken by a patient: often a person knows other languages, e.g., English, that can be easily used to communicate even if nobody knows his native language.

5. Requirement Analysis: The Vending Machine Case Study

To show how the proposed methodology can be generalized, and the fields of application where it achieves the best results, we applied it to another case study belonging to a completely different field: indeed, we consider a company working with *vending machines* and, in particular, we focus on the *machine refill process*, which is at the core of the company business.

In this context, the company wants to add a new, faster and automated channel for the refill requests. Currently, such requests are placed via phone to the company contact center, whose staff compiles a paper form based on the customer requests and manually transmits it to the backoffice staff, which finally creates the corresponding refill order, possibly contacting the customer to ask for further information to uniquely identify the machine to be refilled, and enqueues it in the company sales management system. This process is sketched in Figure 9.

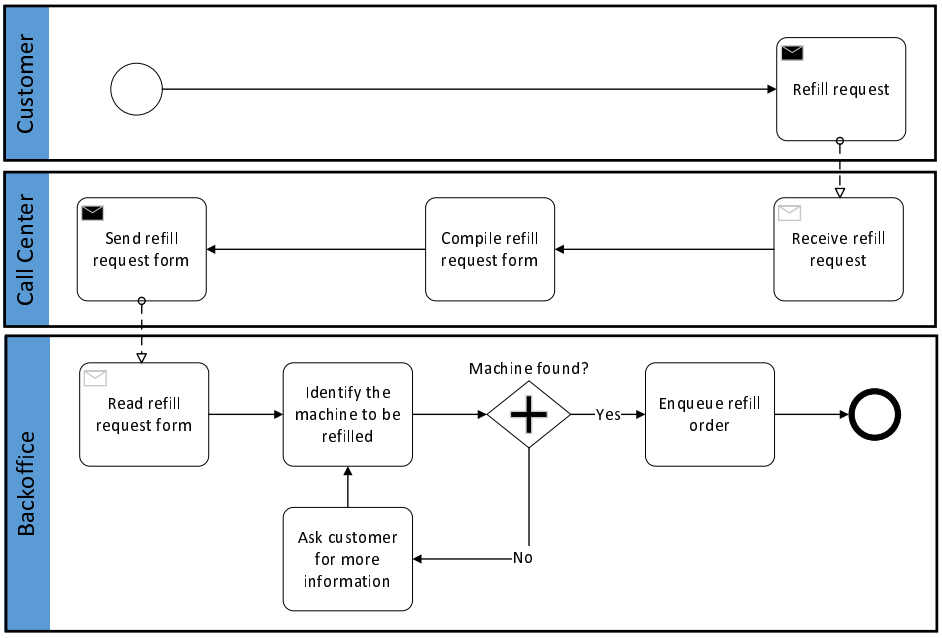

Figure 9. Vending machine refill request: initial BPMN.

Now the company wants the orders to be directly managed by the sales account manager (SAM in the following), which should be able to receive the requests (by email, phone, etc.) and input them in an electronic request form, where the backoffice staff can read it, check it and finally create the order as usual, without any further contact with the customer. In particular, we will focus only on two of the requirements involved by this new process, i.e.,

(A) In order to submit the request, the SAM must identify the specific vending machine to be refilled;
(B) If the customer has no pending payments of up to 100 euros, then the backoffice staff adds the refill request to the order queue.

As in the previous case study, the company IT staff implemented these new requirements following a custom, unstructured integration process, and in both cases there were problems in the requirement analysis phase that led to implementation errors. Thus, we performed a simulation, asking the staff to repeat the requirement integration with the help of our methodology and evaluate the possible advantages deriving from its application.

5.1. Analysis of Requirement A

This requirement tells that the SAM must uniquely identify the vending machine to be refilled (*precondition*) in order to submit the corresponding request (*action*). This is needed to avoid the backoffice contacting the customer, as in the previous process.

Modeling such a (apparently simple) requirement through our iterative methodology allowed us to quickly identify and fix a problem that previously, following the standard analysis process, was caught only during the testing phase.

Indeed the SAM, which was the only stakeholder involved in the previous requirement analysis, initially supposed that the customer name and location were enough to identify a vending machine. However, the backoffice staff, during the first iteration of our process, highlighted that if the customer has more than one machine at the same location, to uniquely identify which machine has to be refilled, it is necessary to specify its *sales point code*. However, the SAM has no direct access to the company software where this information is stored: thus, he may remember it (from a previous request) or ask it to the customer (if possible) or, as the last resort, issue a request to the company headquarters to extract the code from the system.

To model this situation, the next iteration of the process defined a domain model containing a *machine* entity with *customerName*, *customerLocation* and *salesPointCode* attributes, as shown in Figure 10. Here *salesPointCode* is the dominant attribute, which can be set to "asked" (yellow), "unknown" (red) or to any real code (green). The corresponding BPMN is shown in Figure 11.

Entity	Attribute	Description	Value	Color
machine	customerName	the name of the customer this machine belongs to		
	customerLocation	the location where this machine is installed		
	salesPointCode	the sales point code identifying this machine	*unknown*	red
			asked	yellow
			<code>	green

Figure 10. Data model for the "refill request" requirement (A).

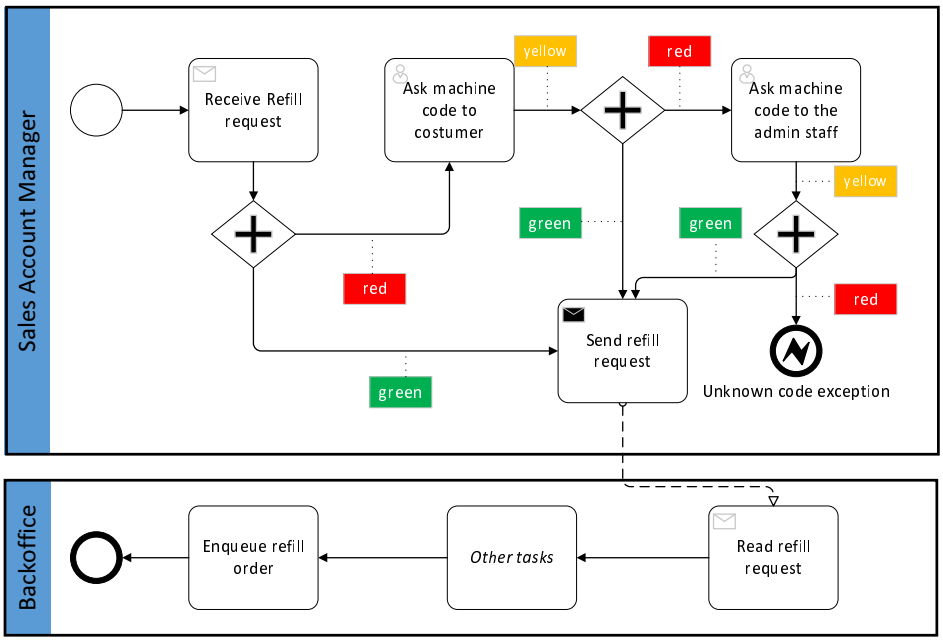

Figure 11. Vending machine refill request: final BPMN for requirement (A).

5.2. Analysis of Requirement B

This requirement tells that a refill request can be placed in the order queue (*action*) by the backoffice staff only if the customer has no more than 100 euros of pending payments (*precondition*).

The precondition refers to the *customer* entity, and in particular to a *pendingPayments* attribute, which is clearly not directly available but is actually the result of a complex query. Therefore, the data model used for this requirement is the one shown in Figure 12, and the resulting BPMN is shown in Figure 13.

Entity	Attribute	Description	Value	Color
customer	pendingPayments	the amount of payments still due	*>100*	red
			<=100	green

Figure 12. Data model for the "enqueue refill order" requirement (B).

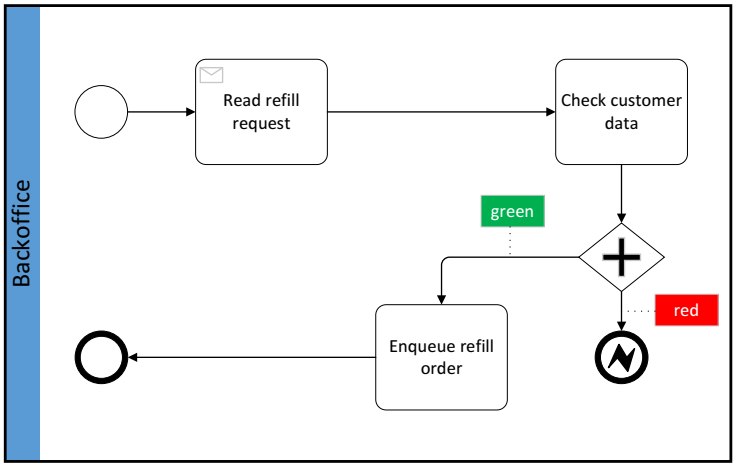

Figure 13. Vending machine refill request: final BPMN for requirement (B).

Here, however, our process turned out to be not detailed enough to achieve a correct implementation, and it encountered the same difficulties as the standard requirement integration methodology previously applied by the company staff. Indeed, the solution above contains a concurrency problem: it is possible for the same request to be handled more than once. Looking at the BPMN and at the data model, it is clear that they do not model the cardinality constraint telling that a request can be managed only once by the staff and, therefore, submitted only once to the order queue. Clearly this problem was not present in the previous paper-based request management process, since paper forms could not be accessed concurrently.

This issue was later fixed by revising the requirement with a new formulation, i.e., "a refill request can be placed in the orders queue by the backoffice staff if and only if the customer has no more than 100 euros in pending payments *and the request has not been already assigned*".

The new part of the requirement precondition refers to a *status* attribute of the *request* entity, which can be "waiting" or "assigned", and is again the result of a query executed on the company management system. Being clearly dominant for the process, the attribute is associated with the yellow (assigned) and blue (waiting) colours as detailed in Figure 14. The corresponding revised BPMN is shown in Figure 15.

Even if our methodology could model the correct solution, it is clear that here it failed, since it did not convey enough information to immediately disclose the concurrency/cardinality issue to the stakeholders involved in the process. In this case, a more detailed data model or class diagram, including constraints and cardinalities, would be helpful, but obviously it would make the requirement definition understandable only to expert stakeholders.

Entity	Attribute	Description	Value	Color
request	status	the refill request staus	*assigned*	yellow
			waiting	blue

Figure 14. Further data model for the "enqueue refill order" requirement (B).

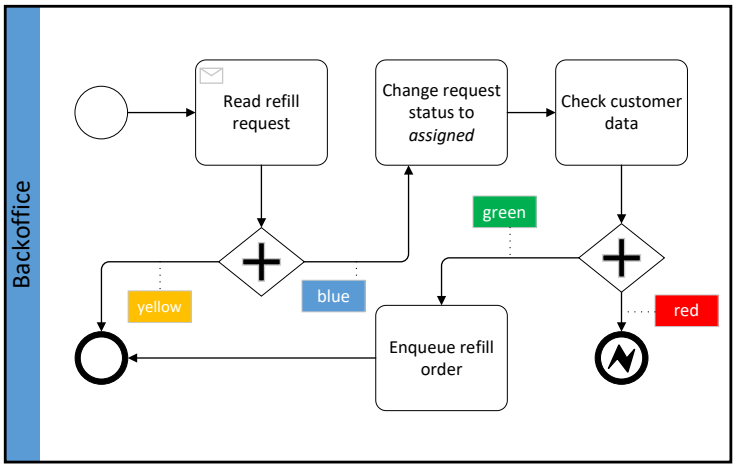

Figure 15. Vending machine refill request: revised BPMN for requirement (B).

6. Discussion

In this section, we discuss key points of the proposed approach, in particular the introduction of the domain data model and the applicability of the methodology to real-world business processes, so to identify the categories of processes that can benefit from its application.

The proposed approach is based on a domain data model that is generated "on demand" from the entities and attributes needed by new or updated requirements being addressed; it aims to eliminate the ambiguity arising from the different terminology used by project stakeholders. The proposed annotation provides a common glossary and introduces a clear separation between the concepts (with their semantics) and their technical implementation, in terms of the details of the underlying data storage and retrieval. Attributes are, in turn, used to create simple conditions on the BMPN model, which are then mapped to visual hints like colored icons, so to easily specify data-driven decision points.

The resulting BPMN model proves to be easier to understand when compared to the use of natural language annotations, which may be ambiguous, too concise (omitting essential details) or too detailed (giving unnecessary information), and/or not easy to read due to excessive text annotations.

A preliminary validation of the proposed approach has been carried out by applying it to real-world case studies, so to evaluate its effectiveness with respect to conventional approaches not using this paper's proposal. The case studies reported in this paper show that, in terms of *applicability*, this paper's contribution *achieves the best results on the integration of requirements that introduce small, noncritical, local updates*, which are actually the most common in the practice. In this case, the implementation of new or updated requirements is carried out by applying several small changes and updates to a specific fragment of the original BPMN model.

On the other hand, when complex, structural modifications to the system are involved, it is often useful to "restart from scratch" and define completely new models for the overall process and for the involved entities. Indeed, in this case, the methodology may not convey enough information to fully model the new requirement and its integration points.

In order to better identify the processes that can be profitably addressed by the proposed methodology, we adopt the process matrix shown Figure 16, based on an original design by [21]. The table classifies processes according to two variables, "process environ-

ment variability" and "output-variation value to customers", thus identifying the following four categories:

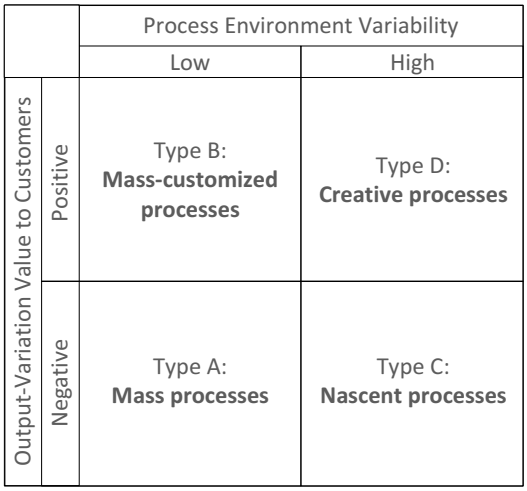

Figure 16. Matrix for process classification (based on Hall and Johnson (2009)).

- A. Mass processes, i.e., standardized processes characterized by no variations of the output;
- B. Mass-customized processes, i.e., standard processes following a protocol to reduce variations in the output;
- C. Nascent processes, i.e., processes involving innovative use of materials, technologies, or designs;
- D. Creative processes, i.e., processes with customer's value variations in the output.

The methodology presented in this paper can be applied to mass processes (type A) since

- The process has low variability, so the system complexity is not high;
- There is multi-actor involvement, which can take advantage of a good requirements engineering approach.

Moreover, the methodology is also suitable for mass-customized processes (type B) since

- The process has, again, low variability;
- Requirements engineering allows us to easily model scenarios with limited customizations with respect to the standard.

On the other hand, nascent processes (type C) and creative processes (type D) are not addressable by the proposed approach, since they involve prototyping and/or complex interaction with stakeholders and even final users.

The two case studies illustrated in Sections 4 and 5 allow one to have a clear impression of the benefits introduced by the proposed approach in terms of effectiveness and versatility, when applied to appropriate processes (i.e., types A and B).

7. The Process and Support Tool

The requirement analysis process described in Section 3 and illustrated in Sections 4 and 5 is supported by a web application.

In particular, to support the process manager in the definition of the initial BPMN fragment, in the web application we adopt the XML interchange format for the BPMN (which can be exported by all of the most used BPMN editing tools) to allow the manager to import an existing BPMN and extract the fragment(s) of interest. In particular, the tool employs the bpmn-js [22] library, which allows one to easily import, modify and display BPMN models on the web (see Figure 17).

Then, the manager has to define the ad hoc data model to be used in the review process. In the web application, this is realized through a simple interface (see Figure 18) that allows us to declare entities and their attributes as key-value pairs. Such attributes

may have only basic types, i.e., string, integer, integer range, float, float range, Boolean, date and time. We also support lists and enumerations on the above types. Each entity, attribute and enumeration value has a mandatory *description* field where the manager must enter a clear description of its intended meaning. A "dominant" checkbox allows us to mark the dominant attributes: in this case, the manager will be able to define a number of subsets of the attribute domain and associate them with a color.

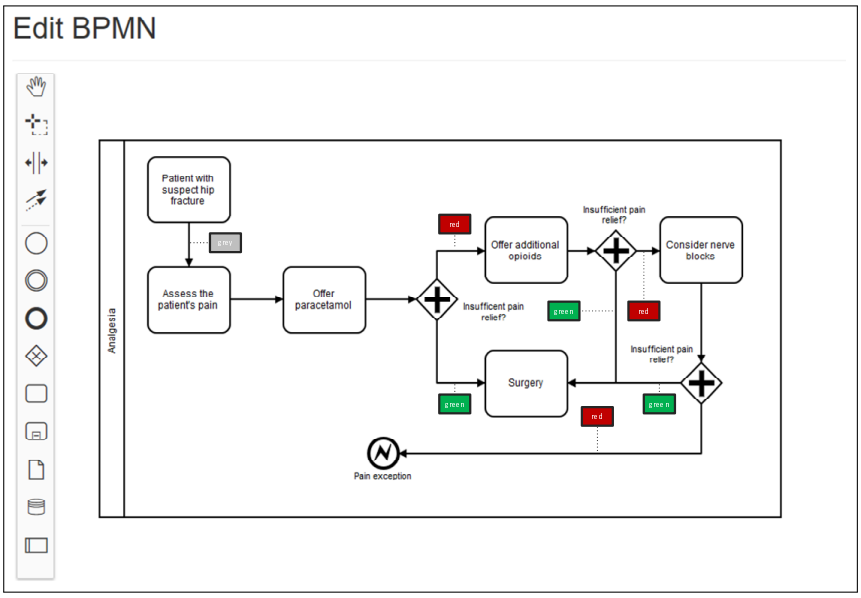

Figure 17. The support tool: BPMN editing.

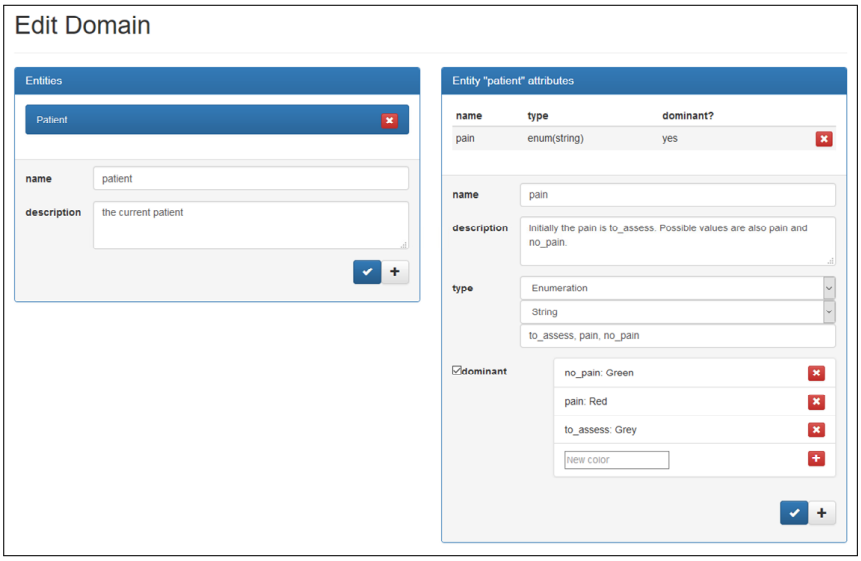

Figure 18. The support tool: domain editing.

Finally, to assist the manager in the formulation of the requirement as a set of pre- and post-conditions placed on the BPMN, the web application exploits the bpmn-js extensibility to allow for the creation of "coloured comments" used to map dominant attributes on the diagram, as described in the previous sections (see again Figure 17). The tool also allows us to enter further comments in natural language in the overall BPMN.

At this point, the tool sends a notification to all the involved stakeholders (defined by the manager), which describes the requirement to integrate and presents a link to enter the review process. After clicking the link, the stakeholder is authenticated and he/she can review the artefacts generated by the manager in the previous steps (BPMN, data model, comments, etc.).

To give feedback, the stakeholders can leave their comments, possibly linking them with specific elements such as process activities or attributes of the data model.

The manager is notified by the system of the stakeholders' feedback and uses it to further refine the artefacts, e.g., updating the BPMN according to the comments.

The process iterates until the manager believes that the stakeholders' comments do not require further changes and/or a specified amount of time is passed without comments.

At the end of the process, the stakeholders' access is disabled and the manager sends all the review artefacts to the developers.

All the process steps above are logged by the web application to a suitable review log that, in particular, keeps track of the comments and of the updates, so the responsibilities of each step are clear.

Note that, given the formal and well-defined structure of the artefacts produced and exploited in the process (i.e, BPMN models, domain data models, etc.), standard model transformation techniques could also be applied to derive other views and artefacts, in order to simplify both the stakeholders' reviews and the developers' implementation tasks (see, e.g., [5]). In particular, thanks to the (mandatory) natural language descriptions attached to all the data model elements, a simple transformation could be exploited to export a textual documentation of the overall process starting from the BPMN, with its annotations suitably linked to the data model, as illustrated above. In this way, a human-readable, non-technical explanation of the revised business process could be produced, which would be useful for explaining it to the actual process actors.

8. Conclusions

This paper has introduced a BPMN-based approach to requirements integration. Specifically, the presented approach introduces a BPMN extension that allows stakeholders and software analysts to easily merge and integrate behavioral and data properties in a BPMN model. The proposed extension is shown to facilitate the interaction among various stakeholders, by offering a common vocabulary of terms and an effective representation of data-driven constraints.

The preliminary validation campaign reported in the paper shows that the methodology can be profitably exploited in contexts characterized by low variability and multi-actor involvement. Future work is going to address a massive experimentation in different application domains, so to further evaluate the approach effectiveness and to consider the opportunity to propose a corresponding "official" BPMN extension.

Author Contributions: Conceptualization, B.I., G.D.P. and A.D.; methodology, B.I.; software, G.D.P.; writing—original draft preparation, B.I., G.D.P. and A.D.; supervision, B.I. 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.

Data Availability Statement: The data has been presented in main text.

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

References

1. Sommerville, I. *Software Engineering*, 10th ed.; Pearson Education: London, UK, 2016.
2. Pressman, R. *Software Engineering: A Practitioner's Approach*; McGraw-Hill: New York, NY, USA, 2009.
3. Maciaszek, L.A. *Requirements Analysis and Systems Design*, 3rd ed.; Addison-Wesley: Boston, MA, USA, 2007.
4. OMG. Business Process Model And Notation (BPMN) Version 2.0. Available online: http://www.omg.org/spec/BPMN/2.0/ (accessed on 15 December 2021).

5. Wei, B.; Delugach, H.S. Transforming UML Models to and from Conceptual Graphs to Identify Missing Requirements. In *Graph-Based Representation and Reasoning, Proceedings of the 22nd International Conference on Conceptual Structures, ICCS 2016, Annecy, France, 5–7 July 2016*; Haemmerlé, O., Stapleton, G., Faron Zucker, C., Eds.; Springer International Publishing: Cham, Switzerland, 2016; pp. 72–79. [CrossRef]
6. Arao, T.; Goto, E.; Nagata, T. "Business process" oriented requirements engineering process. In Proceedings of the 13th IEEE International Conference on Requirements Engineering (RE'05), Paris, France, 29 August 2005; pp. 395–399. [CrossRef]
7. Cardoso, E.C.S.; Almeida, J.P.A.; Guizzardi, G. Requirements engineering based on business process models: A case study. In Proceedings of the 2009 13th Enterprise Distributed Object Computing Conference Workshops, Auckland, New Zealand, 1–4 September 2009; pp. 320–327. [CrossRef]
8. Aysolmaz, B.; Gürsul, M.; Kirchner, K.; Laue, R.; Mertens, R.; Reher, F.; Schönreiter, I.; Turban, B.; Weißbach, R. A reflection on the interrelations between business process management and requirements engineering with an agility perspective. In Proceedings of the 15th International Conference on Business Process Management (BPM 2017), Barcelona, Spain, 10–15 September 2017; pp. 669–680. [CrossRef]
9. Odeh, Y. BPMN in Engineering Software Requirements: An Introductory Brief Guide. In Proceedings of the 9th International Conference on Information Management and Engineering; Association for Computing Machinery, New York, NY, USA, 1 July 2017; pp. 11–16. [CrossRef]
10. Wazlawick, R.S. *Object-Oriented Analysis and Design for Information Systems*; Morgan Kaufmann: Boston, MA, USA, 2014.
11. Highsmith, J. *Agile Software Development Ecosystems*; Addison-Wesley Longman Publishing Co., Inc.: Boston, MA, USA, 2002.
12. Weber, B.; Sadiq, S.; Reichert, M. Beyond rigidity – dynamic process lifecycle support. *Comput. Sci. Res. Dev.* **2009**, *23*, 47–65. [CrossRef]
13. Council of Europe. *Developing a Methodology for Drawing up Guidelines on Best Medical Practice. Recommendation Rec(2001)13 and Explanatory Memorandum*; Council of Europe Publishing: Strasbourg, France, 2002.
14. Unger, A.; Spinola, M.; Pessôa, M. Requirements Engineering approaches to derive Enterprise Information Systems from Business Process Management: A systematic literature review. In Proceedings of the Requirements Engineering und Business Process Management (REBPM) Workshop at Modellierung 2018, Braunschweig, Germany, 21 February 2018.
15. Zarour, K.; Benmerzoug, D.; Guermouche, N.; Drira, K. A systematic literature review on BPMN extensions. *Bus. Process Manag. J.* **2019**, *26*, 1473–1503. [CrossRef]
16. Cardoso, P.; Respício, A.; Domingos, D. riskaBPMN - a BPMN extension for risk assessment. *Procedia Comput. Sci.* **2021**, *181*, 1247–1254. [CrossRef]
17. D'Ambrogio, A.; Paglia, E.; Bocciarelli, P.; Giglio, A. Towards performance-oriented perfective evolution of BPMN models. In *6th International Workshop on Model-Driven Approaches for Simulation Engineering*; Barros, F., Hu, X., Prahofer, H., Denil, J., Eds.; Society for Computer Simulation International: San Diego, CA, USA, 2016; pp. 15:1–15:8.
18. Bocciarelli, P.; D'Ambrogio, A.; Giglio, A.; Paglia, E. A BPMN Extension to Enable the Explicit Modeling of Task Resources. In Proceedings of the 2nd INCOSE Italia Conference on Systems Engineering, Turin, Italy, 14–16 November 2016; pp. 40–47.
19. Scrum Alliance. The Scrum Guide. Available online: http://www.scrumalliance.org/why-scrum/scrum-guide (accessed on 15 December 2021).
20. Pastrana, M.; Ordóñez, H.; Ordóñez, A.; Merchan, L., Requirements Elicitation Based on Inception Deck and Business Processes Models in Scrum. In *Advances in Computing, Proceedings of the 12th Colombian Conference, CCC 2017, Cali, Colombia, 19–22 September 2017*; Solano, A., Ordoñez, H., Eds.; Springer International Publishing: Cham, Switzerland, 2017; pp. 327–339. [CrossRef]
21. Hall, J.M.; Johnson, M. When should a process be art, not science? *Harv. Bus. Rev.* **2009**, *87*, 58–65.
22. Camunda Services GmbH. bpmn-js. Available online: http://bpmn.io/toolkit/bpmn-js/ (accessed on 15 December 2021).