Modeling, Formalizing, and Animating Environment-Aware BPMN Collaborations*

Flavio Corradini[1 [0000−0001−6767−2184]], Luca Mozzoni[1 [0009−0002−9639−0762]], Jessica Piccioni[1 [0009−0003−8513−5894]], Barbara Re[1 [0000−0001−5374−2364]], Lorenzo Rossi[1 [0000−0002−6872−0616]], and Francesco Tiezzi[2 [0000−0003−4740−7521]]

[1] University of Camerino, Computer Science Division, Camerino, Italy
{name.surname}@unicam.it
[2] Dipartimento di Statistica, Informatica, Applicazioni, University of Florence, Italy
francesco.tiezzi@unifi.it

Abstract. Business processes, in particular collaborations, describe how various participants interact and behave to achieve specific objectives. Depending on the business scenario, process participants operate in a specific environment characterized by spatial and contextual dimensions. Participants can interact with and modify the environment, which in turn may influence process execution. Indeed, there exists a bidirectional relationship between business processes and the environment, which involves the necessity of representing the environment in a way that allows business processes to benefit from its awareness. Despite extensive research on environment modeling, the seamless integration of business processes and environment models has not been explored yet. To address this gap, we conceptualize environment-aware BPMN collaboration models to capture spatial and contextual dimensions at the desired level of granularity and abstraction; we formalize the operational semantics of such models; and we propose a tool for animating their execution with the aid of geographic maps. We illustrate our findings through an emergency response collaborative scenario.

Keywords: Environment-Aware BPMN · Formal Semantics · Animation.

1 Introduction

Business processes specify how different participants behave, exchange information, and interact to reach an objective. In this regard, the BPMN collaboration [20] is the most adopted notation to model business processes with different participants. Business processes usually happen in a physical space, which is also indicated with the term *environment* [6]. The environment comprises spatial dimensions, which describe metric properties of the tangible space occupied and navigated by process participants. The extent of a corridor connecting two rooms is an example of such spatial dimensions. Beyond these, the environment is also characterized by invisible contextual dimensions that capture domain knowledge [23]. Such dimensions influence how the environment is perceived and utilized, revealing interactions beyond its spatial dimensions. The decreasing of the speed limit in a residential area to ensure safety is an example of such

* This work was partially supported by projects SERICS (PE00000014) and VITALITY (ECS_00000041) under the NRRP MUR - NextGenerationEU, and by MUR issue D.M. 118/2023 "Doctoral programs of national interest in Blockchain & Distributed Ledger Technology".

contextual dimensions. The ability to leverage such environmental dimensions, both spatial and contextual, is referred to as *environmental awareness* [8].

Within the scope of BPMN, environmental awareness enables participants to optimize operations on the basis of the working environment [22,24]. A process participant can hold a position or move between locations, e.g., move toward a parking lot; check the environment's state, e.g., check for free parking slots; and modify it, e.g., occupy a parking slot. In parallel, the environment can influence tasks and decisions of a process, e.g., a traffic light that temporarily stops a movement task or the temperature of a room that drives the decision taken by a gateway. Considering collaborations, the environment acts as a shared resource, providing a common context that participants access and interact with, influencing each other's behavior. For instance, participants can move synchronously in the environment, e.g., a taxi driver who carries a passenger; can take possession of environmental resources, e.g., a participant who occupies a toilet, making it inaccessible to others; or can indirectly communicate by modifying the environment and responding to these modifications (aka stigmergy), e.g., a participant planting tomatoes, thereby enabling another participant to detect and water the area accordingly. Overall, such interplay establishes a bidirectional relationship between the collaboration and the environment, where one influences the other and vice versa.

Therefore, an accurate representation of the environment, properly integrated with collaborations to achieve its awareness, are crucial for correctly managing business processes [22]. In this regard, environment modeling has been largely studied in literature [2,3,12,18,14,13]. Concerning environmental awareness in BPMN, most approaches focus solely on extending its graphical notation, while only a few works [24,6] define environment models integrated with business process models. However, all these works discard relevant aspects: the possibility to model different parts of the same environment at different levels of *granularity*, thus enabling environmental reasoning with distinct accuracy; the possibility to define different levels of *abstraction*, thus enabling the seamless integration of high-level environmental aspects in the collaboration model; the possibility to specify *constraints on movements* involving different participants, thus enabling the representation of situations where a participant follows the same movements of another. Finally, these works lack approaches to support the modeler in the specification of the environment and the visualization of process execution on it.

To this aim, we propose an approach to model, formalize, and animate environment-aware BPMN collaborations, whose effectiveness and feasibility are demonstrated through an emergency case study. The contribution of our work is threefold. We first provide a **conceptualization of environment-aware BPMN collaboration models** integrating BPMN collaborations with an environment model that captures both spatial and contextual dimensions, allowing multiple levels of granularity. Specifically, we extend BPMN along two directions. (i) We define new BPMN task elements that explicitly represent movements in the environment and binding/unbinding participants to enable synchronized movements within the same environment. (ii) We define an environment model based on a *semantically-enriched place graph*, whose nodes represent locations in the space, edges define how locations are connected, and both places and edges are equipped with *attributes* representing either spatial or contextual dimensions. Exploiting attributes, we can define in our environment model different *logical views*, provid-

ing higher levels of abstraction with respect to the physical layer described by the place graph. Each view consists of a set of logical places defined by expressions on attributes. Logical places are equipped in their own turn with logical attributes, whose values can be get/set by means of specific *aggregation/disaggregation functions*. We then define a **formalization** of the conceptualized environment-aware BPMN collaboration models, which rigorously describes the execution of a BPMN collaboration on its working environment. Finally, we present a **tool for animating and debugging** environment-aware BPMN collaborations. The tool implements the formal semantics to guarantee a faithful execution of the model. The tool graphically represents the place graph on top of the geographic map from which it has been derived. Thus, the model execution is visualized over both the BPMN model and the geographic map.

The rest of this paper is organized as follows. Sec. 2 discusses environment modeling. Sec. 3 introduces environment-aware BPMN collaborations through a running example. Sec. 4 presents the relevant aspects of our formal semantics. Sec. 5 illustrates our animation tool. Sec. 6 presents related works. Finally, Sec. 7 concludes the paper.

2 Environment Modeling

Classifications of models representing the environment in a structured manner to capture its relevant properties have been largely addressed in previous research [2,3,12]. The literature emphasizes that a model's specific properties define the extent to which environmental reasoning can be applied. For instance, shortest path queries may be less precise or even infeasible to perform, depending on the aspects captured by the model, highlighting the strong interconnection between its design and intended use. Models are classified into *geometric* and *symbolic* [2]. Geometric models consider space as continuous or discrete, usually via *cell-based* and *boundary-based* representations. These models typically focus on quantitative aspects of space, such as room dimensions or road lengths, providing precise information. However, they require an integration of semantic annotations to achieve a higher degree of environmental awareness. Conversely, symbolic models focus on qualitative aspects, such as connectivity and containment, mostly utilizing *set-based* and *graph-based* representations. Symbolic models facilitate environmental awareness by providing human-readable descriptions, though they are typically less accurate due to the lack of geometric details. From an application perspective, symbolic-based models are often preferred over geometric-based ones, as they can capture the semantics of entities and places. All the above models encode spatial *topology* by representing connectivity among locations, an essential feature for navigational reasoning. However, none captures all the relevant aspects necessary to achieve full environmental awareness. To overcome this limitation, hybrid models [18,15,10] provide combinations between geometrical and symbolic information. In particular, hybrid models, such as *semantically-enriched place graphs*, are commonly used in practice.

Another key aspect to consider is the level of detail at which information is represented, often referred to as *granularity*, which determines the environmental reasoning accuracy [17]. A coarser granularity groups larger areas, like entire floors or neighborhoods, while a finer granularity captures more precise distinctions, such as individual rooms within a building. However, finer granularity increases the number of elements, which in turn decreases query performance. The impact on performance

depends on the model's scalability, i.e., its ability to maintain efficient environmental reasoning as the number of elements grows. To this aim, some models adopt *abstraction mechanisms* to support reasoning over large environments. Hierarchical models, for instance, manage multiple abstraction layers effectively [10,4], but require inter-level connections and detailed geometry, increasing overall complexity [2]. Granularity is key to balancing the model's accuracy and performance. Nevertheless, it depends on whether the environment is indoor or outdoor. Indoor spaces typically feature more complex layouts and constrained movement paths, often requiring a finer granularity. To address this, some studies have explored unified modeling approaches [14,13,26], enabling the usage of multiple granularities within the same model. Table 1 resumes the key characteristics of the presented environment models. Guided by these key aspects, we decided to adopt semantically-enriched place graphs as they enable the representation of spatial topology and the annotation of their elements with both symbolic and geometric properties. As a result, the spatial extent covered by each element can be individually defined, thus supporting multiple granularities. Moreover, abstraction mechanisms can be defined on top of the model without relying on complex hierarchical constructs.

Features \ Paper	**[18]**	**[15]**	**[4]**	**[10]**	**[13]**	**[14]**	**[26]**
Topology	•	•	•	•	•	•	•
Symbolic annotations	•	•	•	•			•
Geometric annotations	•	•	•	•	•	•	•
Multiple granularities	•	•	•		•	•	•
Abstraction mechanisms			•	•			

Table 1: Resuming environment models.

3 Environment-aware BPMN Collaboration

This section introduces a running example, explores how to model the environment, and establishes its connection to processes through an environmental BPMN meta-model.

Running Example. We adopt a simple running example set in a university compound. The environment is composed of two buildings, *Building A* and *Building B*, and a parking lot between them. The collaboration concerns two participants: a *Student* and a *Tutor Buddy*. Since the student needs course assistance, they visit the tutor's office. The tutor buddy uses the university mobile app to find an available study room, guiding the student there and providing information before they part ways.

Modeling the Environment. To achieve environment-aware BPMN collaborations, it is essential to represent relevant spatial and contextual dimensions through an environment model, which is then combined with BPMN collaborations. Specifically, the environment is modeled as a *semantically-enriched place graph* whose nodes and edges represent locations and their reachability, respectively. Both places and edges can be characterized by attributes, representing either spatial or contextual dimensions, which process participants can interpret to make decisions. Spatial attributes enable reasoning about place extents and distances, such as computing the shortest path between two classrooms. Contextual attributes extend this reasoning to additional dimensions, such as determining the least crowded path rather than just the shortest one. However, operating in large environments may require process participants to reason at different scales, particularly when combining both indoor and outdoor spaces. For example, we may want to determine whether the student is inside one of the two buildings or in a specific classroom within them. To manage these variations, the model must seamlessly

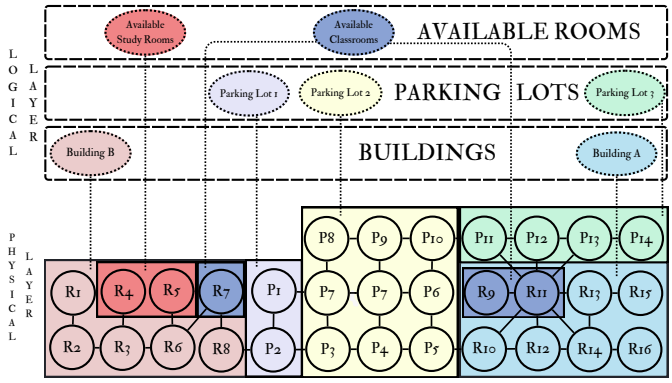

Fig. 1: Running example: environment model.

integrate multiple abstract views, providing different levels of abstraction. To this aim, we resort to an environment model that defines a logical layer on top of the place graph, which we now refer to as the physical layer. The logical layer is made of two main components: *logical places* and *views*. A logical place refers to a set of physical places within the place graph; each logical place is defined through an expression on the attributes of physical places. A view is instead the union of one or more logical places and their logical attributes, whose values are given by the application of specific functions.

Figure 1 represents our approach applied to the running example, illustrating one possible way to model its environment. The physical layer includes places representing the rooms in the two buildings, labeled from *R1* to *R16*, and the parking slots, from *P1* to *P15*. Such a physical layer of the place graph can be obtained by tessellating the space into homogeneous areas. Also, each physical place has a *zone* contextual attribute indicating its main area, and places in the two buildings also have *freeSeats* and *purpose* contextual attributes, specifying available seats and their purpose of use. The logical layer contains seven logical places and three views. The logical place *Available Study Rooms* is defined by all physical places that satisfy the expression "$purpose == studying$ and $freeSeats > 0$", while the logical place *Available Classrooms* by the expression "$purpose == teaching$ and $freeSeats > 30$". In *Available Study Rooms*, a room is available if it has at least one free seat, as these spaces are meant for individual study. While *Available Classrooms* require 30 free seats, as they are reserved as whole units. Similarly, the logical places *Building A* and *Building B* include all physical places with the *zone* attribute equal to *A* or *B*, respectively. Finally, the *Parking Lot 1*, *Parking Lot 2*, and *Parking Lot 3* logical places correspond to physical places where the *zone* attribute has value *park 1*, *park 2*, or *park 3*. Concerning the views, *Available Rooms* is formed by combining two logical places: *Available Study Rooms* and *Available Classrooms*. An aggregation function is then used to dynamically compute the value of their attribute *seats*, based on the values of the attribute *freeSeats* in the corresponding physical places. Similarly, a disaggregation function can be used to set the attribute *seats* to occupy free seats in the underlying physical places. The *Buildings* view contains *Building A* and *Building B* logical places, and the *Parking Lots* view contains the *Parking Lot 1*, *Parking Lot 2*, and *Parking Lot 3* logical places.

However, one may argue that parking lots are not crucial in the running example, as they merely serve as transit areas between buildings. Thus, they could be represented

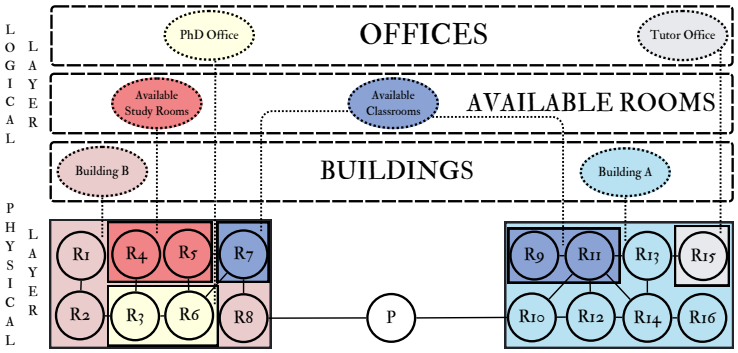

Fig. 2: Running example: adjusted environment model.

with a coarser granularity by using a single physical place *P* to cover the entire parking area. Additionally, a new view can be added to deal with offices of tutors and PhDs, see Figure 2. By allowing flexibility in granularity, the model can be adjusted by omitting irrelevant details. This emphasizes how modeling the two layers depends on the specific use case, requiring the modeler to make thoughtful decisions accordingly.

Combining Environment with BPMN Meta-model. To capture environmental aspects, we extend the BPMN meta-model using its extensibility mechanism [25], as described in Figure 3. Notably, all generalization associations in the metamodel are assumed to be disjoint and incomplete; for the sake of readability, their labels have been omitted in Figure 3. For what concerns the environment, we introduce two new classes for the physical layer, called *Physical Place* and *Edge*. *Physical Place* refers to *Edge* (and vice-versa) through two attributes: *sourcePlace* and *targetPlace*. The *Physical Place* and *Edge* classes also have a list of environmental attributes called *envAttribute*, which refers to spatial and contextual dimensions of the environment. For the logical layer, we introduce the class *View*, which is a composition of one or more elements of the class *Logical Place* and their environmental attributes. Each view also has a list of aggregation and disaggregation functions, which offer a transparent mechanism for accessing and updating logical place attributes, based on the values of the corresponding physical place environmental attributes. For the sake of presentation, from now on, we use the term attribute to refer to an environmental attribute. The connection between the environment and a BPMN collaboration happens through the attribute *position*, which links the BPMN classes *Participant* and *Physical Place*. This relationship is used to indicate the current position of participants in the environment. Another connection between the environment and a BPMN collaboration lies in tasks as they may depend from the environment and affect it. BPMN process tasks have two new attributes, *assignment* and *guard*, used to modify and constrain the execution to the environment's state, respectively. These are defined through expressions, which indeed may refer to places and their attributes. In the same way, as XOR gateways specify conditions through expressions, the decisions they take may also depend on the environment's state. To explicitly model movement, we introduce the class *Movement Task*, which extends the BPMN class *Task* and represents tasks involving movement within the environment. These tasks specify a *destination* attribute, which corresponds to a *Place*: an abstract class extended by both the *Logical Place* and *Physical Place* classes. A movement task is graphically distinguished by an arrow icon. Nevertheless, a participant's movement

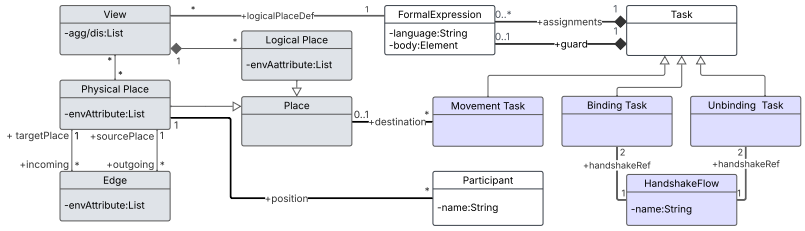

Fig. 3: Environmental BPMN Meta-model.

is not always self-directed. In collaborative scenarios, one participant may define the destination for another, revealing the need for synchronization between them. To handle this, the concept of *handshake* is introduced, representing an interaction where two participants synchronize their movements within the same environment. To model this interaction, we introduce two new task type classes: *Binding Task* and *Unbinding Task*. The former establishes a link between participants, enabling one to follow the movements of another. In contrast, the latter breaks this connection, allowing participants to resume independent movements and actions. These tasks can only be executed if the two participants are in the same location in the environment. To distinguish them from other types of tasks, binding tasks graphically report a handshake icon, while unbinding tasks report a two-separated hand icon. Additionally, the two types of tasks are connected with a new flow called *HandshakeFlow*, represented by a continuous line. As indicated in the metamodel, the HandshakeFlow associations indicate that one flow connects a maximum of two binding or unbinding tasks. Using multiple HandshakeFlows, it is possible to bind (or unbind) an arbitrary number of participants. For instance, if participant A binds with participant B, and subsequently binds with participant C, then a movement by A will result in all three participants moving together. Notably, undesirable behaviors may occur during the execution due to modeling errors. For instance, if two bound participants move simultaneously, it becomes unclear which one should be followed, resulting in a non-deterministic behavior. Anyway, such modeling errors are detected and notified by the animation tool introduced in Section 5.

To give a concrete view of the newly introduced elements that form environment-aware BPMN collaborations, we provide in Figure 4 the BPMN collaboration diagram representing the participants' behavior of the running example. The collaboration starts with the student performing a movement task to reach the tutor's office for assistance. Notably, the destination of this movement task is specified as the logical place *Tutors Office*. The usage of logical places, instead of physical places, in the BPMN collaboration model, permits to simplify the modeling task, as the modeler can refer to the space in the environment by means of logical names (e.g., *Tutors Office*) rather than low-level identifiers (e.g., $R1$). In addition, this permits decoupling the collaboration model from the physical layer of the environment, so that the same collaboration model continues to behave correctly even if the environment model is reconfigured (e.g., the tutors' office is relocated to a different physical place). The tutor receives the assistance request, registers it into a data object, and brings the student to an available study room to assist him. Also, the task to move to the study room specifies a logical place as its destination, i.e. *Available Student Room*; however, this logical place belongs to a different view that permits to reason in terms of seat availability. Before moving, the two

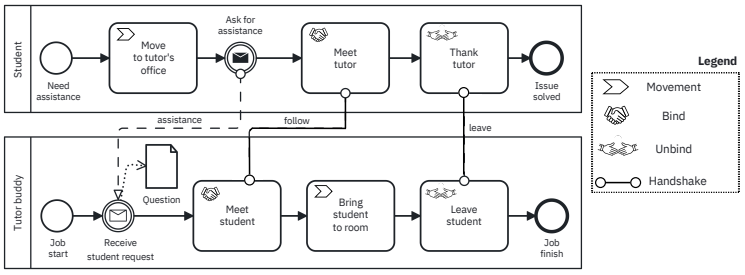

Fig. 4: Student-Tutor collaboration process.

participants perform a handshake. Indeed, the student should not go where he/she wants but has to follow the tutor. This is represented in the model by the binding tasks *Meet tutor* and *Meet student* and the unbinding tasks *Leave student* and *Thank tutor*. Binding tasks enable the student to move in sync with the tutor until they reach the designated available room, where they then perform unbinding tasks.

4 Formal Account of Environment-aware Collaborations

This section provides the formal semantics of environment-aware BPMN collaborations. The formalization extends the one proposed in [6] with new BPMN elements, specifically devised for dealing with the interplay between the collaboration processes and the environment. Most of all, our formalization considers a richer model of the environment. To keep the formalization manageable and understandable, we focus on elements that are strictly needed to define meaningful models. For the sake of presentation, we following provide the essential part of the formalization, for a more detailed account of it the interested reader can refer to the companion technical report [1].

Formal Syntax. The formalization resorts to a textual representation that uses a Backus-Naur Form (BNF) syntax for representing the BPMN collaboration model and the environment model. The motivation for using here a textual representation rather than the usual graphical notation is that the former is more manageable for writing operational semantic rules than the latter. In addition, the textual notation makes explicit those technical details of collaboration models that are not part of the graphical representation (e.g., message payloads, assignments, guard conditions), but are part of the low-level XML characterization of the model. This information is needed to properly define the execution semantics of the environment-aware BPMN collaboration models.

Figure 5 reports the BNF syntax defining the textual notation describing the structure of environment-aware BPMN collaboration models. Specifically, the upper table reports the grammar productions defining the syntax of collaboration models together with the notation of the related generic elements of the syntactic categories. Symbol $*$ stands for (possibly empty) sequences, and $+$ for non-empty sequences. As usual in BPMN, we assume that message flows, sequence flows, and task have unique names in the model; we make the same assumption for handshake flows names. The bottom table reports the definition of the environment, together with the related notation.

Intuitively, a environment-aware BPMN collaboration is the union of the terms C, indicating the structure of the collaboration, and ENV, indicating the environment. A collaboration is rendered as a collection of pools coupled with an environment model

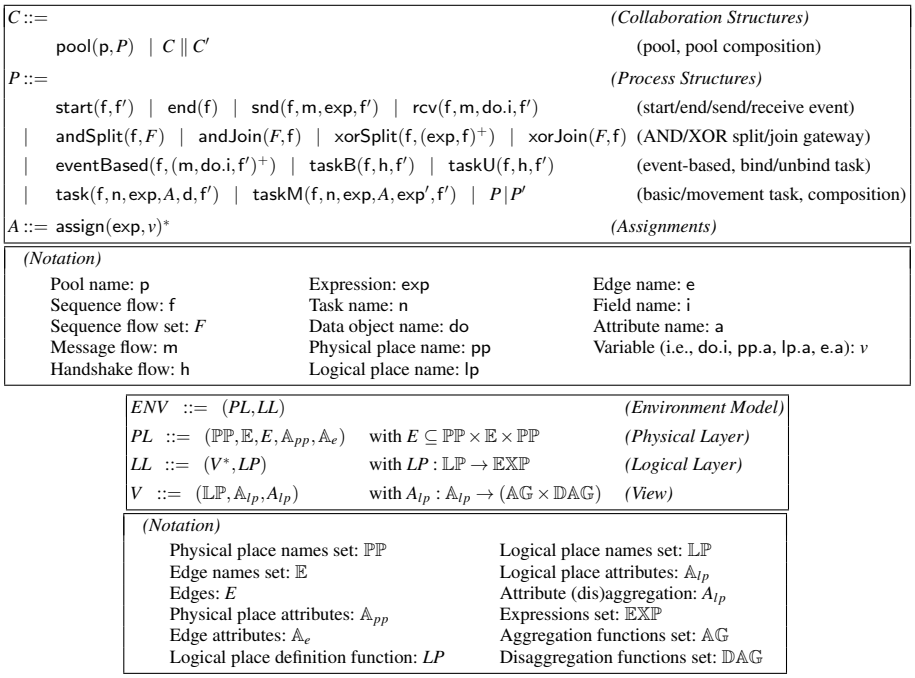

Fig. 5: BNF syntax of environment-aware BPMN collaboration models.

composed of a physical layer and a logical layer. Formally, a collaboration C is a composition by means of the operator $\|$, of pool elements $\mathsf{pool}(\mathsf{p}, P)$ uniquely identified by a pool name p and enveloping process structures of the form P. Similarly, a process P is a composition of process elements (denoted by the sans serif font) through the operator $|$. To support a compositional approach, each sequence/message flow of the graphical notation is split into two parts: the part outgoing the source element and the part incoming the target element. The two parts are correlated by a unique sequence/message flow name. The correspondence between the textual terms of the process elements and the usual graphical ones is straightforward. We describe following the meaning of tasks, which have a key role in the interplay with the environment; the meaning of the other elements is standard. The term $\mathsf{task}(\mathsf{f}, \mathsf{n}, \mathsf{exp}, A, \mathsf{d}, \mathsf{f}')$ denotes a basic task, which has an incoming sequence flow f, a name n, a guard exp that is a conditional expression predicating on fields and attributes, a list of assignments A, a duration d that represents the amount of time it takes to execute the task, and an outgoing sequence flow f'. An assignment $\mathsf{assign}(\mathsf{exp}, v)$ assigns the value resulting from the evaluation of the expression exp to a variable v, which can be a data object field $\mathsf{do.i}$, an attribute of a physical place $\mathsf{pp.a}$, an attribute of a logical place $\mathsf{lp.a}$, or an attribute of an edge $\mathsf{e.a}$. For the sake of readability, when the guard and/or the assignments are not used, they are omitted from the syntactic definition of the element. $\mathsf{taskM}(\mathsf{f}, \mathsf{n}, \mathsf{exp}, A, \mathsf{exp}', \mathsf{f}')$ denotes a movement task, which is a type of task that may involve a movement in the environment; it is similar to the basic task, but has an additional expression exp' that represents the destination to reach. In addition, a movement task does not have the term d, because it is assumed that its duration is given by the time to reach the destination. $\mathsf{taskB}(\mathsf{f}, \mathsf{h}, \mathsf{f}')$ denotes a binding task that, besides incoming and outgoing sequence flows f and f', specifies

$$
\begin{array}{rcl}
C_{univ} & = & \mathsf{pool}(\mathsf{p}_{student}, P_{student}) \parallel \mathsf{pool}(\mathsf{p}_{tutor}, P_{tutor}) \\
P_{student} & = & \mathsf{start}(\mathsf{f}_1', \mathsf{f}_1) \mid \mathsf{taskM}(\mathsf{f}_1, \mathsf{n}_{moveToTutorOffice}, \mathsf{lp}_{tutorOffice}, \mathsf{f}_2) \mid \mathsf{snd}(\mathsf{f}_2, \mathsf{m}_{assistance}, \mathsf{v}_{question}, \mathsf{f}_3) \mid \\
 & & \mathsf{taskB}(\mathsf{f}_3, \mathsf{h}_{follow}, \mathsf{f}_4) \mid \mathsf{taskU}(\mathsf{f}_4, \mathsf{h}_{leave}, \mathsf{f}_5) \mid \mathsf{end}(\mathsf{f}_5) \\
P_{tutor} & = & \mathsf{start}(\mathsf{f}_6', \mathsf{f}_6) \mid \mathsf{rcv}(\mathsf{f}_6, \mathsf{m}_{assistance}, \mathsf{do}_{question}.\mathsf{i}_{msg}, \mathsf{f}_7) \mid \mathsf{taskB}(\mathsf{f}_7, \mathsf{h}_{follow}, \mathsf{f}_8) \mid \\
 & & \mathsf{taskM}(\mathsf{f}_8, \mathsf{n}_{bringStudentToRoom}, (\mathsf{lp}_{availableStudyRooms}.\mathsf{a}_{seats} > 0), \\
 & & \qquad \mathsf{assign}((\mathsf{myplace}.\mathsf{a}_{freeSeats} - 1), \mathsf{myplace}.\mathsf{a}_{freeSeats}), \mathsf{lp}_{availableStudyRooms}, \mathsf{f}_9) \mid \\
 & & \mathsf{taskU}(\mathsf{f}_9, \mathsf{h}_{leave}, \mathsf{f}_{10}) \mid \mathsf{end}(\mathsf{f}_{10}) \\
\hline
ENV_{univ} & = & (PL_{univ}, LL_{univ}) \\
PL_{univ} & = & (\{\mathsf{pp}_{R1}, \ldots, \mathsf{pp}_{R16}, \mathsf{pp}_P\},\ \{\mathsf{e}_1, \ldots, \mathsf{e}_{20}\},\ \{(\mathsf{pp}_{R1}, \mathsf{e}_1, \mathsf{pp}_{R2}), (\mathsf{pp}_{R2}, \mathsf{e}_1, \mathsf{pp}_{R1}), \ldots, (\mathsf{pp}_{R16}, \mathsf{e}_{20}, \mathsf{pp}_{R14})\}, \\
 & & \{\mathsf{a}_{zone}, \mathsf{a}_{purpose}, \mathsf{a}_{freeSeats}\},\ \{\mathsf{a}_{status}\}) \\
LL_{univ} & = & (V_{buildings}, V_{availableRooms}, V_{offices}, LP_{univ}) \\
LP_{univ} & = & [\mathsf{lp}_{buildingA} \mapsto (\mathsf{a}_{zone} == \mathsf{v}_A),\ \mathsf{lp}_{buildingB} \mapsto (\mathsf{a}_{zone} == \mathsf{v}_B), \\
 & & \mathsf{lp}_{availableStudyRooms} \mapsto (\mathsf{a}_{purpose} == \mathsf{v}_{studying} \text{ and } \mathsf{a}_{freeSeats} > 0), \\
 & & \mathsf{lp}_{availableClassrooms} \mapsto (\mathsf{a}_{purpose} == \mathsf{v}_{teaching} \text{ and } \mathsf{a}_{freeSeats} > 30), \\
 & & \mathsf{lp}_{phdOffice} \mapsto (\mathsf{a}_{purpose} == \mathsf{v}_{research}),\ \mathsf{lp}_{tutorOffice} \mapsto (\mathsf{a}_{purpose} == \mathsf{v}_{tutoring})] \\
V_{buildings} & = & (\{\mathsf{lp}_{buildingA}, \mathsf{lp}_{buildingB}\}, \emptyset, \emptyset) \\
V_{availableRooms} & = & (\{\mathsf{lp}_{availableStudyRooms}, \mathsf{lp}_{availableClassrooms}\}, \{\mathsf{a}_{seats}\}, [\mathsf{a}_{seats} \mapsto (sum\langle \mathsf{a}_{freeSeats} \rangle, occupy\langle \mathsf{a}_{freeSeats} \rangle)]) \\
V_{offices} & = & (\{\mathsf{lp}_{phdOffice}, \mathsf{lp}_{tutorOffice}\}, \emptyset, \emptyset)
\end{array}
$$

```
sum⟨a⟩(this):                    occupy⟨a⟩(this,n):
  res := 0;                        foreach pp in 𝒫(this):
  foreach pp in 𝒫(this):             if (n > pp.a):  n -= pp.a;  pp.a := 0;
    res += pp.a;                     else:  pp.a -= n;  break;
  return res;
```

Fig. 6: Textual representation of the running example.

a handshake flow h that connects it to the binding task of another participant (which has the same h). $\mathsf{taskU}(\mathsf{f}, \mathsf{h}, \mathsf{f}')$ denotes an unbinding task, which specifies a handshake flow h that connects it to the unbinding task of another participant (which has the same h). Tasks' expressions can use the distinguished place myplace to refer to the current physical place of their participant.

Let us focus now on the syntax of the environment model. An environment ENV is a pair (PL, LL) whose elements represent two layers. The physical layer PL consists of a place graph, i.e., a graph whose nodes represent physical places (whose names are in the set $\mathbb{PP}$) and directed arcs represent edges E (whose names are in the set $\mathbb{E}$) connecting places. Both physical places and edges are equipped with contextual information represented in terms of attributes (whose names belong to the sets $\mathbb{A}_{pp}$ and $\mathbb{A}_e$, respectively). The logical layer LL consists of a set of views and a function $LP : \mathbb{LP} \to \mathbb{EXP}$ that specifies which physical places form each logical place. Notably, the physical places forming a logical place are not statically listed in the model, because the constituents of a logical place may change dynamically. Thus, the LP function maps each logical place to a boolean expression on physical place attributes, whose evaluation on a physical place determines if it belongs (at the time of such evaluation) to the corresponding logical place. Formally, the set of physical places forming a given logical place $\mathsf{lp} \in \mathbb{LP}$ is defined as follows: $\mathscr{P}(\mathsf{lp}) = \{\mathsf{pp} \in \mathbb{PP} \mid [LP(\mathsf{lp})]_{\mathsf{pp}}\}$, where function $[\cdot]_{\mathsf{pp}}$ returns an expression resulting from the instantiation of the input expression with the physical place name pp, i.e., $[\mathsf{exp}]_{\mathsf{pp}}$ replaces each attribute a in exp with pp.a. Finally, a view V groups a set of logical places $\mathbb{LP}$, defining how the values of their attributes (in the set $\mathbb{A}_{lp}$) result from the aggregation of the values of physical place attributes and, vice versa, propagate from the logical to the physical layer. The handling of the logical place attributes of a view is defined by the function A_{lp} that maps each logical attribute to an aggregation and a disaggregation function. The aggregating function of a logical attribute is

called during the evaluation of an expression when the logical attribute is encountered; the disaggregating function, instead, is called when an assignment, having the logical attribute as target, is processed. An aggregation function always returns the aggregated value, while a disaggregation function returns void. In practice, these functions provide a transparent mechanism to access and update the value of a logical attribute (like a C# property). This allows the modeler to simply deal with logical attributes as standard physical attributes in expressions and assignments, thus relieving him/her from the duty of explicitly handling aggregation and disaggregation of values. In Figure 6 we provide the textual representation of the running example, whose BPMN collaboration model is depicted in Figure 4 and its environment in Figure 2. This exemplifies the correspondence between the graphical representation of the model and the proposed BNF syntax. Note that in Figure 2, edges are undirected, so they are represented here by two directed edges with the same edge name, thus having the same attribute values. At the bottom of Figure 6, we report the definition, in terms of pseudocode, of the aggregation function *sum* and the disaggregation function *occupy*.

Formal Semantics. To describe the semantics of environment-aware BPMN collaboration, we enrich the structural part of the model with the execution state (denoted by σ). These stateful descriptions are called *process configurations*, with the form $\langle P, ENV, \mathsf{p}, \sigma \rangle$, and *collaboration configurations*, with the form $\langle C, ENV, \sigma \rangle$. A state σ consists of a list of *state functions*, among which we mention here: the *sequence flow state function* σ_f specifying for each sequence flow the current number of tokens marking it; the *physical place attribute state function* σ_{ppa} assigning values to physical place attributes; the *handshake state function* σ_h that, given a participant p, returns the set of participants with which p is bound; the *task state function* σ_s used to keep track of the active tasks; the *place state function* σ_p mapping each participant to a physical place. The state obtained by updating in σ_f the number of tokens of the sequence flow f to n, written as $\sigma_f \cdot [\mathsf{f} \mapsto \mathsf{n}]$, is defined as follows: $(\sigma_f \cdot [\mathsf{f} \mapsto \mathsf{n}])(\mathsf{f}')$ returns n if $\mathsf{f}' = \mathsf{f}$, otherwise it returns $\sigma_f(\mathsf{f}')$. The update of the other state functions is similarly defined.

The semantics of environment-aware BPMN collaborations is given in the structural operational semantics style [21] by relying on the notion of *Labeled Transition System* (LTS). Specifically, we define an LTS for the process behavior and another for the collaboration behavior, whose states represent process and collaboration configurations, respectively. The labels of LTSs are: a binding label $\rightarrow \mathsf{h} \leftarrow$; an unbinding label $\leftarrow \mathsf{h} \rightarrow$; a silent action τ (due, e.g., to token flow in the processes); and a timed action $\checkmark$ denoting the passing of a unit of time (due, e.g., to movements in the environment). The transition relations of the LTSs are the least relations induced by a set of operational rules. Indeed, they define the transitions from one configuration to another. Each rule is designed to capture the behavior of a specific BPMN element. The most significant rules are given in Figure 7; the rest is given in the companion technical report [1].

To improve the readability, when possible, we omit: the environment *ENV* and the the execution state σ from the source configuration of transitions; the process/collaboration structure and the environment from the target configuration of transitions; those state functions from target configurations that are not affected by transitions. The names of the operational rules are aligned with the kinds of elements they operate on; for instance, the rule named *TaskM* deals with movement tasks. Moreover, to further simplify

$$\mathsf{taskM}(\mathsf{f},\mathsf{n},\mathsf{exp}_g,A,\mathsf{exp}_d,\mathsf{f}') \xrightarrow{\tau} \langle dec(\sigma_f,\mathsf{f}), enab(\sigma_s,\mathsf{n})\rangle \quad \sigma_f(\mathsf{f})>0 \wedge \sigma_s(\mathsf{n})=dis \wedge eval(\mathsf{exp}_g) \quad (TaskM_A)$$

$$\mathsf{taskM}(\mathsf{f},\mathsf{n},\mathsf{exp}_g,A,\mathsf{exp}_d,\mathsf{f}') \xrightarrow{\checkmark} \langle move(\sigma_p,\sigma_h,\mathsf{p},\mathsf{pp})\rangle \quad \sigma_s(\mathsf{n})=enab \wedge \mathsf{pp} \in next(PL,\sigma_p,\mathsf{p},eval(\mathsf{exp}_d)) \quad (TaskM_{TI})$$

$$\mathsf{taskM}(\mathsf{f},\mathsf{n},\mathsf{exp}_g,A,\mathsf{exp}_d,\mathsf{f}') \xrightarrow{\checkmark} \langle\rangle \quad \sigma_s(\mathsf{n})=dis \quad (TaskM_{T2})$$

$$\mathsf{taskM}(\mathsf{f},\mathsf{n},\mathsf{exp}_g,A,\mathsf{exp}_d,\mathsf{f}') \xrightarrow{\checkmark} \langle\rangle \quad \sigma_s(\mathsf{n})=enab \wedge next(PL,\sigma_p,\mathsf{p},eval(\mathsf{exp}_d))=\emptyset \wedge \neg arrived(\sigma_p,\mathsf{p},eval(\mathsf{exp}_d)) \quad (TaskM_{T3})$$

$$\mathsf{taskM}(\mathsf{f},\mathsf{n},\mathsf{exp}_g,A,\mathsf{exp}_d,\mathsf{f}') \xrightarrow{\tau} \langle inc(\sigma_f,\mathsf{f}'), upd(A), dis(\sigma_s,\mathsf{n})\rangle \quad \sigma_s(\mathsf{n})=enab \wedge arrived(\sigma_p,\mathsf{p},eval(\mathsf{exp}_d)) \quad (TaskM_C)$$

$$\mathsf{taskB}(\mathsf{f},\mathsf{h},\mathsf{f}) \xrightarrow{\rightarrow \mathsf{h} \leftarrow} \langle inc(dec(\sigma_f,\mathsf{f}),\mathsf{f}')\rangle \quad \sigma_f(\mathsf{f})>0 \quad (TaskB)$$

$$\frac{P_1 \xrightarrow{\rightarrow \mathsf{h} \leftarrow} \langle\sigma'\rangle \quad \langle P_2,\sigma'\rangle \xrightarrow{\rightarrow \mathsf{h} \leftarrow} \langle\sigma''\rangle \quad \sigma_p(\mathsf{p}_1)=\sigma_p(\mathsf{p}_2)}{\mathsf{pool}(\mathsf{p}_1,P_1) \parallel \mathsf{pool}(\mathsf{p}_2,P_2) \xrightarrow{\tau} \langle \sigma'' \cdot bind(\sigma_h,\mathsf{p}_1,\mathsf{p}_2)\rangle} \quad (C\text{-}Bind)$$

Fig. 7: Environment-aware BPMN operational semantics: excerpt of rules.

Function $inc(\sigma_f,\mathsf{f}) = \sigma_f \cdot [\mathsf{f} \mapsto \sigma_f(\mathsf{f})+1]$, resp. $dec(\sigma_f,\mathsf{f}) = \sigma_f \cdot [\mathsf{f} \mapsto \sigma_f(\mathsf{f})-1]$, increments, resp. decrements, by one the number of tokens marking the sequence flow f in the state σ_f.
Function $enab(\sigma_s,\mathsf{n}) = \sigma_s \cdot [\mathsf{n} \mapsto enab]$, resp. $dis(\sigma_s,\mathsf{n}) = \sigma_s \cdot [\mathsf{n} \mapsto dis]$, activates, resp. deactivates, the task n in σ_s.
Function $eval(ENV,\sigma_{doi},\sigma_{ppa},\sigma_{ea},\sigma_p,\mathsf{exp}) = \mathsf{v}$ states that v is the value resulting from the evaluation of the expression exp on the data fields in σ_{doi}, attributes in σ_{ppa} and σ_{ea}, and participants' positions in σ_p. For logical place attributes, the evaluation resorts to the aggregation functions specified in the views of the environment model ENV. We omit the environment and the state functions when they are clear from the context, thus writing $eval(\mathsf{exp})$.
Function $upd(ENV,\sigma_{doi},\sigma_{ppa},\sigma_{ea},\sigma_p,A)$ performs the assignments A and returns the updated data object, physical place attribute and edge attributes state functions. The assignment of a logical place attribute $\mathsf{lp.a}$ resorts to the disaggregation function corresponding to a in the view containing lp. specified in the environment model ENV. We omit the environment and the state functions when they are clear from the context, thus writing $upd(A)$.
Function $next(PL,\sigma_p,\mathsf{p},\mathsf{pp})$ returns the set of places that are next in the shortest paths in the physical layer PL from the current place $\sigma_p(\mathsf{p})$ of the process participant to the destination pp; the function returns $\emptyset$ when there is no path to the destination. $next(PL,\sigma_p,\mathsf{p},\mathsf{lp})$ returns $next(PL,\sigma_p,\mathsf{p},\mathsf{pp})$ where $\mathsf{pp} \in \mathscr{P}(\mathsf{lp})$.
Function $arrived(\sigma_p,\mathsf{p},\mathsf{pp})$ returns $(\sigma_p(\mathsf{p}) == \mathsf{pp})$, which is *true* if the current position of the participant p is the destination pp. Instead, $arrived(\sigma_p,\mathsf{p},\mathsf{lp}) = (\sigma_p(\mathsf{p}) \in \mathscr{P}(\mathsf{lp}))$.
Function $move(\sigma_p,\sigma_h,\mathsf{p},\mathsf{pp})$ assigns to the the participant p, and all participants bound to it, the physical place pp in the state σ_p. Formally, $move(\sigma_p,\sigma_h,\mathsf{p},\mathsf{pp}) = \sigma_p \cdot [\mathsf{p} \mapsto \mathsf{pp}] \cdot [\mathsf{p}_1 \mapsto \mathsf{pp}] \cdot \ldots \cdot [\mathsf{p}_n \mapsto \mathsf{pp}]$ with $\sigma_h(\mathsf{p}) = \{\mathsf{p}_1,\cdots,\mathsf{p}_n\}$.
Function $bind(\sigma_h,\mathsf{p}_1,\mathsf{p}_2)$ binds the participants p_1 and p_2 in σ_h. Formally, $bind(\sigma_h,\mathsf{p}_1,\mathsf{p}_2) = \sigma_h'$ with $\sigma_h'(\mathsf{p}) = \sigma_h(\mathsf{p})$ if $\mathsf{p} \notin \{\mathsf{p}_1,\mathsf{p}_2\}$, $\sigma_h'(\mathsf{p}_1) = \sigma_h(\mathsf{p}_1) \cup \{\mathsf{p}_2\}$, and $\sigma_h'(\mathsf{p}_2) = \sigma_h(\mathsf{p}_2) \cup \{\mathsf{p}_1\}$. Function *unbind* is similarly defined, where $\cup$ is replaced by $\setminus$. Notation $\sigma \cdot \sigma_h'$ denotes the update of state σ by replacing the enclosed σ_h by σ_h'.

Fig. 8: Auxiliary functions.

the definition of the operational rules, in Figure 8 we define *auxiliary functions* that update the state functions of process configurations or perform checks in rules' conditions.

We now briefly comment on the rules in Figure 7. Rule $TaskM_A$ is responsible for activating movement tasks. It applies when there is a token marking its incoming sequence flow ($\sigma_f(\mathsf{f}) > 0$), the task is currently disabled ($\sigma_s(\mathsf{n}) = dis$), and the guard exp_g is satisfied (predicate $eval(\mathsf{exp}_g)$ returns true). Upon activation, the rule consumes the token by removing it from the incoming sequence flow ($dec(\sigma_f,\mathsf{f})$) and enables the task ($enab(\sigma_s,\mathsf{n})$). Referring to the running example, this rule is applied to the movement task with name $\mathsf{n}_{bringStudentToRoom}$ and, in that case, the guard that checks if there are some free seats in the logical places related to available study rooms ($\mathsf{lp}_{availableStudyRooms} \cdot \mathsf{a}_{seats} > 0$) evaluates true. To evaluate the logical attribute a_{seats} within the expression, *eval* resorts to the first element of the pair (given by the projection function $\downarrow_1$) returned by the A_{lp} function belonging to the view $(\mathbb{LP}, \mathbb{A}_{lp}, A_{lp})$ in ENV such that $\mathsf{lp}_{availableStudyRooms} \in \mathbb{LP}$. In this case, $A_{lp}(\mathsf{a}_{seats}) \downarrow_1 = sum\langle \mathsf{a}_{freeSeats} \rangle$; thus, the aggregated value of the attribute is returned by the call

$sum\langle \mathsf{a}_{freeSeats}\rangle(\mathsf{lp}_{availableStudyRooms})$. Notably, during the execution of the aggregation function code, the value of physical attribute $\mathsf{pp.a}$ is get by $\sigma_{ppa}(\mathsf{pp},\mathsf{a})$. Rule $TaskM_{T1}$ performs a movement to the next place of the physical layer, and it applies when the task is enabled and the next physical place is reachable. Rule $TaskM_{T2}$ enables movement tasks to evolve with temporal transitions when they are disabled. Rule $TaskM_{T3}$ enables movement tasks to evolve with temporal transitions when it is enabled, the destination is not reachable, and the current position is not the destination. Rule $TaskM_C$ deals with the completion of the movement task; if the task is enabled and the destination has been reached, the rule increments the outgoing sequence flow, performs the assignments, and disables the task. Assuming the rule is applied to a movement task specifying the assignment $\mathsf{assign}(5, \mathsf{lp}_{availableStudyRooms}.\mathsf{a}_{seats})$, used to reserve seats on study rooms. Function *upd* is used to perform the assignment; it resorts to the second element of the pair returned by A_{lp}. In this case, $A_{lp}(\mathsf{a}_{seats}) \downarrow_2 = occupy\langle \mathsf{a}_{freeSeats}\rangle$. The update of the execution state is then computed by the call $occupy\langle \mathsf{a}_{freeSeats}\rangle(\mathsf{lp}_{availableStudyRooms}, 5)$. Notably, during the execution of the disaggregation function code, an assignment $\mathsf{pp.a} := \mathsf{exp}$ corresponds to $\sigma_{ppa} \cdot [(\mathsf{pp},\mathsf{a}) \mapsto eval(\mathsf{exp})]$. Rule *TaskB* deals with binding tasks, producing a transition label $(\rightarrow \mathsf{h} \leftarrow)$ about the will of the participant to bind with the other one connected by the handshake flow h. Rule *C-Bind* binds two participants if they are in the same physical place. Similar rules deal with unbinding.

5 Tool and Evaluation

We now present our web application for animating environment-aware BPMN collaborations and discuss its features through a case study. The tool repository, available at `https://zenodo.org/records/15587686` and `https://pros.unicam.it/environmental-bpmn/`, contains source code, a user guide, and 11 case studies spanning different application domains. Most are provided in two variants—intended behavior and error scenarios—to showcase debugging features. Some case studies focus on binding and unbinding tasks, such as those in the transportation domain. Others, like those in the agriculture domain, emphasize the dynamicity of logical places and environmental attributes, and share the same environment model.

Environment-aware BPMN Collaborations Animator. The tool's interface, shown in Figure 9, displays the BPMN collaboration on the left (highlighted in green), the environment model in the center (highlighted in red), and a right-side panel listing environment's spatial and contextual dimensions (highlighted in light blue). The BPMN collaboration is modeled using the integrated BPMN modeler, which comprises the additional elements previously introduced. The environment is modeled in JSON with the assistance of coordinate extrapolation tools, such as `https://www.keene.edu/campus/maps/tool/`, enabling its overlay on a geographic map.

The tool's main feature is process animation, which begins when the play button (top-left) is clicked and ends when no tokens can move. Colored tokens (one color per participant) indicate execution progress, flowing through the BPMN and environment models to represent the process state and participant positions. The animation can be paused at any time, allowing users to inspect the token distribution across the two models and assess the environment's state through the right-side panel. This panel allows users to inspect attributes of places, both physical and logical, and edges, and track their

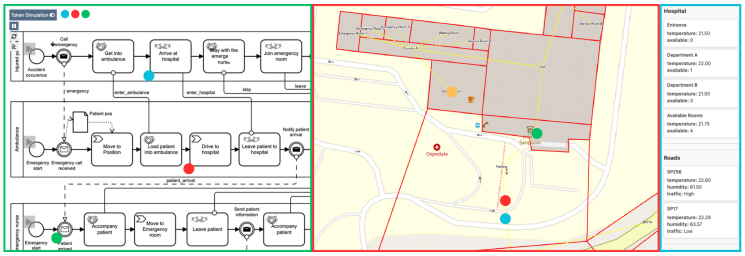

Fig. 9: Environment-aware BPMN collaborations animator GUI.

evolution over time. It lists both contextual and spatial attributes, with spatial attributes also being visualized on the map, such as by coloring the extent of a place.

During the animation, warning and error messages support effective debugging of environment-aware BPMN collaborations by enabling the designers to identify undesired executions. Warnings indicate potential deadlock situations, i.e. two participants attempting a binding task from different positions, a participant unable to reach a specific destination, or a violated guard. Animation continues, as these conditions may eventually resolve. Errors, however, occur when the collaboration cannot proceed, i.e. a missing participant's initial position or two bound participants moving in opposite directions, and cause the animation to stop.

Emergency response case study. We present a case study that models an emergency response collaboration within a hospital and its surroundings. The collaboration concerns four participants: *Injured Patient*, *Ambulance*, *Emergency Nurse*, and *Emergency Doctor*. It starts when the *Injured Patient* is involved in a car accident and calls an ambulance, providing its position. Upon receiving the call, the *Ambulance* moves to the patient's position. During this operation, the ambulance's travel can be seen in both models: its process token waits near the movement task while its environment token moves across the map. However, if the accident occurs in an unreachable location, such as due to a closed road, the tool displays a warning message, depicted in the upper part of Figure 10. Similarly, failing to specify the ambulance's initial position displays an error message, depicted in the lower part of Figure 10, and halts the animation. The collaboration continues with the ambulance picking up and transporting the patient to the hospital. From a process perspective, the tokens wait for each other near the respective binding tasks. When bound, the patient's token waits at the next unbinding task, while the ambulance executes the movement task to return to the hospital. About the environment, both tokens move across the map, with the ambulance dictating the patient's movements. Once at the hospital, the ambulance's token reaches an unbinding task, detaching from the patient.

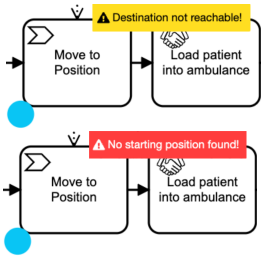

Fig. 10: Debugging features.

6 Related Work

In the literature, several works focus on the interplay between business processes and the environment. Decker et al. [7], Grefen et al. [11] and Zhu et al. [27] propose BPMN

extensions that allow modeling location-based tasks whose execution is constrained by the participant's location; in addition, the latter introduce location-dependent gateways whose conditions are based on environmental information. Similarly, Mazhar et al. [19] propose a new type of element to represent participant movements and group task elements based on the location where they are performed. Differently, Dorndorfer et al. [9] discuss the use of environmental information to trigger conditional boundary events and thus handle exceptional behaviors. Similarly, Chiu et al. [5] introduce a new type of event called location event triggered by environmental conditions captured by IoT sensors. Also, Poss et al. [22] adopt location-based events and, in addition, define location data, i.e., data objects used to store and retrieve environmental information, and introduce the possibility of allocating resources to tasks based on the location. Tomas Kozel [16] does a step forward, proposing, in addition to location-based events, the possibility to use specific markers for participants that can move in the environment and track their position. In our work, we include all these concepts adding environmental constraints on tasks and gateways and representing movements explicitly with movement tasks. Differently from the mentioned works, we introduce novel tasks that bind or unbind participants, enabling synchronized movements within the same environment. We also combine an environment model with BPMN collaborations and formalize it.

	Features \ Paper	**[7]**	**[11]**	**[27]**	**[19]**	**[9]**	**[5]**	**[22]**	**[16]**	**[24]**	**[6]**	**our**
Environment	Physical layer										•	•
	Logical layer											•
	Contextual attributes										•	•
	Spatial attributes											•
	Multiple granularities											•
	Abstract views											•
	Agg/Dis functions											•
BPMN Model	Location-based task	•	•	•	•			•		•	•	•
	Movement task				•						•	•
	Binding/Unbinding task											•
	Location-based gateway			•						•	•	•
	Location-based events					•	•	•	•		•	•
	Participant position								•	•	•	•
Formal semantics										•	•	•
Tool support												•

Table 2: Resuming related work.

Concerning the combination of an environment model with BPMN, only a few works consider it. Saddem-Yagoubi et al. [24] extend and formalize with temporal logic the BPMN notation. They introduce location-based gateways and location-based tasks and provide a graph-based environment model without any graphical representation. Nevertheless, they primarily focus on how the environment affects process execution while neglecting the impact of process activities on the environment, which instead is our main focus. Corradini et al. [6] introduce a model capable of representing the interplay between BPMN and the environment. The adopted environment model represents the physical layer of the environment with a fixed granularity without the possibility of specifying logical layers, which is one of the key contributions of our work. Still differently from our work, the environment model in [6] captures only contextual attributes, neglecting spatial ones, and does not account for interactions where participants synchronize their movements within the same environment. Additionally, the authors consider movement tasks using standard tasks without providing an explicit definition. Our approach also overcomes the above works by formalizing and representing large-scale environments, both physically and logically, at different levels of granularity, along with their spatial and contextual attributes, supported by aggregation and disaggregation functions. In this way, our environment-aware BPMN collaboration model seamlessly integrates multiple abstract views, providing different levels of abstraction. Moreover, our work is the only one that provides a tool to easily animate and

debug the integrated model. Summing up, even though the literature lately is increasingly dealing with the concept of environment within BPM, our contribution overcomes the current state of the art for different aspects, as resumed in Table 2.

7 Concluding Remarks

This paper explores the concept of environment and its interplay with business processes to expose the need for environmental awareness. To enable it, BPMN collaboration model is integrated with an explicit model of the environment, which proves to be fundamental for reasoning on spatial and contextual attributes, and on relationships between places and edges (e.g., reachability), information not captured by raw business data. Having a separate model for the environment also supports reuse across scenarios and allows for graphical representation, as witnessed by our animation tool.

Discussion. In this paper, we focus on the conceptualization of environment-aware BPMN collaboration models, deliberately working at an abstract level and avoiding low-level implementation concerns. While achieving a realistic environment-aware collaboration would require integration with sensors capable of tracking the location and movement of people or objects, our goal is to provide a conceptual abstraction. In line with this choice, we also avoid working with raw geographical information, as it would introduce complexity and irrelevant details at the level of modeling we target. Indeed, we propose an abstraction by leveraging semantically-enriched place graphs that allows for a simpler specification of the environment, which is more suitable for a seamless integration at business process level and more manageable for a formal treatment.

We use a reduced version of the BPMN notation. Including such additional BPMN elements as event sub-processes or boundary events would allow us to deal with anomalous or exceptional situations following an event-driven style. In addition, while the proposed approach supports the modeling of interactions among multiple participants through the use of separate pools, each representing an individual participant, we acknowledge that scalability can be further improved by incorporating multi-instance pools. However, the above extensions would also have significantly increased the complexity of the approach, both in terms of formalization and implementation.

It is worth noticing that the approach to build environment-aware BPMN collaboration models may vary depending on the domain and context of use. In fact, in some cases the environment model is known a priori and the process model has to be developed on top of it, while in others the process is already available and it has to be deployed and checked in different environments. For this reason, we leave it up to the modeler to decide how to proceed to build environment-aware BPMN collaborations.

Future Works. As future work, we plan to explore how the enactment of environment-aware BPMN collaborations can be achieved by integrating an environment modeler into the tool and enabling interoperability with external sources. In particular, we aim to incorporate spatial modeling capabilities and leverage data from sensors to construct a logical environment grounded in real-world data. Moreover, we plan to extend our approach with additional BPMN elements, enabling a broader range of modeling scenarios, including the handling of exceptions and multiple instances. Finally, we intend

to exploit the formal semantics and its implementation, to enable the verification of properties using, e.g., model checking techniques.

References

1. Technical report, https://tinyurl.com/techreportBPM2025
2. Afyouni, I., Ray, C., Christophe, C.: Spatial models for context-aware indoor navigation systems: A survey. Journal of Spatial Information Science **1**(4), 85–123 (2012)
3. Becker, C., Dürr, F.: On location models for ubiquitous computing. Personal and Ubiquitous Computing **9**, 20–31 (2005)
4. Cagigas, D., Abascal, J.: Hierarchical path search with partial materialization of costs for a smart wheelchair. Journal of Intelligent and Robotic Systems **39**, 409–431 (2004)
5. Chiu, H.H., Wang, M.S.: Extending event elements of business process model for internet of things. In: CIT/IUCC/DASC/PICOM. pp. 783–788. IEEE (2015)
6. Corradini, F., et al.: On the Interplay Between BPMN Collaborations and the Physical Environment. In: BPM. LNCS, vol. 14940, p. 93–110. Springer (2024)
7. Decker, M., et al.: Modeling mobile workflows with BPMN. In: ICMB-GMR. pp. 272–279. IEEE (2010)
8. Dobson, S., et al.: Spatial awareness in pervasive ecosystems. KER **31**(4), 343–366 (2016)
9. Dörndorfer, J., Seel, C.: A framework to model and implement mobile context-aware business applications. In: Modellierung 2018, pp. 23–38. Gesellschaft für Informatik e.V. (2018)
10. Fernández, J.A., González, J.: Multi-hierarchical representation of large-scale space: Applications to mobile robots. Springer Science & Business Media (2001)
11. Grefen, P., Brouns, N., Ludwig, H., Serral, E.: Co-location specification for IoT-aware collaborative business processes. In: CAiSE. LNCS, vol. 350, pp. 120–132. Springer (2019)
12. Hu, H., Lee, D.L.: Semantic location modeling for location navigation in mobile environment. In: Mobile Data Management. pp. 52–61. IEEE (2004)
13. Hussein, S.H., Lu, H., Pedersen, T.B.: Towards a unified model of outdoor and indoor spaces. In: Advances in Geographic Information Systems. pp. 522–525. ACM (2012)
14. Jensen, S.K., et al.: Outdoor-indoor space: Unified modeling and shortest path search. In: Indoor Spatial Awareness. pp. 35–42. ACM (2016)
15. Jiang, C., Steenkiste, P.: A hybrid location model with a computable location identifier for ubiquitous computing. In: Ubiquitous Computing. pp. 246–263. Springer (2002)
16. Kozel, T.: BPMN mobilisation. In: ECCS. pp. 307–310. WSEAS (2010)
17. Kuhn, W., Ballatore, A.: Designing a language for spatial computing. In: AGILE. pp. 309–326. LNGC, Springer (2015)
18. Lorenz, B., Ohlbach, H.J., Stoffel, E.: A hybrid model for indoor spatial reasoning. In: GAR Workshop. pp. 2–7. Citeseer (2006)
19. Mazhar, S., Wu, P., Rosemann, M.: Designing complex socio-technical process systems – the airport example. BPMJ **25** (11 2018)
20. OMG: Business process model and notation. (BPMN V2.0) (2011)
21. Plotkin, G.D.: A structural approach to operational semantics. JLAMP **60-61**, 17–139 (2004)
22. Poss, L., Schönig, S.: Location-aware business process modeling and execution. SoSyM pp. 1–31 (2024)
23. Rosemann, M., et al.: Contextualisation of business processes. IJBPIM **3**(1), 47–60 (2008)
24. Saddem-Yagoubi, R., Poizat, P., Houhou, S.: Business processes meet spatial concerns: the sbpmn verification framework. In: Formal Methods. pp. 218–234. Springer (2021)
25. Stroppi, L.J.R., Chiotti, O., Villarreal, P.D.: Extending BPMN 2.0: Method and Tool Support. In: Business Process Model and Notation, pp. 59–73. Springer (2011)
26. Worboys, M.: Modeling indoor space. In: ISA workshop. pp. 1–6 (2011)
27. Zhu, X., et al.: Exploring location-dependency in process modeling. BPMJ (2014)