Resilient BPMN: Robust Process Modeling in Unreliable Communication Environments

Frank Nordemann[1], Ralf Tönjes[1] and Elke Pulvermüller[2]
[1]*Faculty of Engineering and Computer Science, Osnabrück University of Applied Sciences, Osnabrück, Germany*
[2]*Institute of Computer Science, University of Osnabrück, Osnabrück, Germany*
{f.nordemann, r.toenjes}@hs-osnabrueck.de, elke.pulvermueller@informatik.uni-osnabrueck.de

Keywords: Business Process Modeling, Language Extension, Meta Modeling, Unreliable Communication Environments, Dynamic Process Adaption, Process Robustness Verification, BPMN, DMN, QoS, OppNets, DTNs, *rBPMN*.

Abstract: Process modeling languages help to define and execute processes and workflows. The Business Process Model and Notation (BPMN) 2.0 is used for business processes in commercial areas such as banks, shops, production and supply industry. Due to its flexible notation, BPMN is increasingly being used in non-traditional business process domains like Internet of Things (IoT) and agriculture. However, BPMN does not fit well to scenarios taking place in environments featuring limited, delayed, intermittent or broken connectivity. Communication just exists for BPMN - characteristics of message transfers, their priorities and connectivity parameters are not part of the model. No backup mechanism for communication issues exists, resulting in error-prone and failing processes. This paper introduces *resilient BPMN (rBPMN)*, a valid BPMN extension for process modeling in unreliable communication environments. The meta model addition of opportunistic message flows with Quality of Service (QoS) parameters and connectivity characteristics allows to verify and enhance process robustness at design time. Modeling of explicit or implicit, decision-based alternatives ensures optimal process operation even when connectivity issues occur. In case of no connectivity, locally moved functionality guarantees stable process operation. Evaluation using an agricultural slurry application showed significant robustness enhancements and prevented process failures due to communication issues.

1 INTRODUCTION

Process Modeling Languages (PML's) are used for the definition of business processes and workflows. Examples include traditional flow charts, UML Activity diagrams and event-driven process chains from ARIS (Dumas et al., 2018). However, the de-facto standard is BPMN 2.0: due to its flexibility, expressiveness and usability for domain experts, its non-proprietary and extensible design, mature tool support and ISO-Certification (ISO, 2013), BPMN fits many different use cases of business processes. Even in non-traditional business domains such as IoT and agriculture, BPMN is a promising choice for modeling and executing process definitions (cf. Section 2).

In terms of communication, BPMN 2.0 may not fulfill the requirements of a domain perfectly though. Communication between actors/system parts can be realized by message flows in collaboration diagrams. Specific views for the cooperation between actors/system parts can be modeled using choreography and conversation diagrams. However, communication properties such as available bandwidth, latency and failure probability are not in the focus of BPMN process modeling and execution. This is a problem for scenarios in which solid and faultless communication does not exist. Unreliable communication environments feature limited, delayed, intermittent or broken connectivity in dynamically changing network topologies. Many scenarios combine infrastructure-based (e.g. cellular, WiFi in access-point mode) and infrastructure-free (ad-hoc) communication technologies, forming hybrid networks based on node's connectivity capabilities and mobility characteristics. In particular, this applies to Opportunistic Networks (OppNets) and Delay Tolerant Networks (DTNs) (Fall, 2003), (Pelusi et al., 2006). Examples for unreliable communication environments include use cases in developing and rural areas (e.g. agriculture, road construction, wildlife observation), IoT scenarios with connectivity limited devices, use cases in untrustworthy infrastructure and disaster scenarios.

Modeling of non-functional communication aspects and alternatives for failing connectivity is difficult and inflexible in BPMN. Mechanisms for on-demand process adaption based on connectivity parameters are missing. Robustness issues are identified not before process execution fails at runtime.

This paper introduces *resilient BPMN (rBPMN)*, a domain specific modeling language for unreliable communication environments. Using this valid BPMN extension, domain experts may model processes and verify their robustness in graphically understandable, not overloaded collaboration diagrams. The main research contributions of the paper are

1. a lightweight BPMN meta model extension for opportunistic message flows,
2. a mechanism for process robustness verification at design time,
3. methods for explicit and implicit, dynamic opportunistic alternatives and decision taking for optimal process operation and
4. a method for moveable functionality in case of no connectivity.

An example for an unreliable communication environment is provided by the agricultural domain. Agriculture takes place in rural and sparsely populated areas using a small amount of mobile machines. Communication between machines, farms, personnel and Internet services is often limited or broken.

The paper starts with related work (Section 2), followed by the presentation of *rBPMN* (Section 3) with its domain requirements, equivalence check, the domain extension model and robustness verification. Next is a use case featuring an agricultural slurry application in Section 4 and a comparison to robust process modeling in plain BPMN 2.0 in Section 5. Conclusion and future work is presented in Section 6.

2 RELATED WORK

BPMN 2.0 has been extended in various ways since its definition by the Object Management Group (OMG) in 2011 (OMG, 2011). BPMN was designed following a model driven approach. Its metamodeling architecture defines the structure, meaning and behavior of objects and is based on the Meta Object Facility (MOF) of OMG (OMG, 2014). The integrated extension mechanism allows to extend the modeling language for special requirements of new use cases and domains. However, according to (Braun and Esswein, 2014a) less than 20 percent of available extensions implement the provided mechanism. This is probably caused by syntactical and methodical misunderstandings as argued by (Braun, 2015). Valid extensions enable model interchangeability and remain compatible with the core of BPMN, an important aspect when using existing tools and runtime engines. A selection of extensions is presented subsequently.

Numerous extensions provide solutions to integrate resources (Stroppi et al., 2011a), (Braun and Esswein, 2014b), (Bocciarelli et al., 2016), (Betke and Seifert, 2017) and performance criteria (Bocciarelli and D'Ambrogio, 2011), (Bocciarelli et al., 2014) into BPMN.

Many publications present solutions for an integration of IoT and Cyber Physical Systems (CPS) into the BPMN meta model. Work often concentrates on concepts for physical entities, sensors and actuators, adding new task types, events and attributes to the meta model (Meyer et al., 2013), (Meyer et al., 2015), (Graja et al., 2016), (Bocciarelli et al., 2017). Quality of Information (QoI) of physical resources is focused in (Martinho and Domingos, 2014).

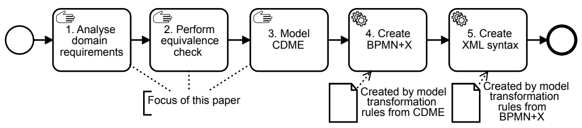
QoS in web service process models and SOA-based service discovery is part of (Bocciarelli and DAmbrogio, 2014). (Mazzola et al., 2017) provide a cloud-based environment with compensations for faulty tasks and QoS-based process optimization. In both publications, QoS is referring to non-functional aspects such as performance, workload and reliability of individual process tasks and resources. QoS-aspects like bandwidth, latency and failure probability of the actual transport network are not considered. Furthermore, the solutions do not address unreliable communication scenarios and robustness verification.

A couple of publications address process, task and resource reliability (Bocciarelli et al., 2014), (Respício and Domingos, 2015), (Domingos et al., 2016). Other extensions include mobility (Kozel, 2010), clinical pathways (Braun et al., 2014), composite applications (Kopp et al., 2012), mobile context (Dörndorfer and Seel, 2017) and ubiquitous computing (Yousfi et al., 2016).

To the best of the authors' knowledge, none of the available extensions focuses on robust process modeling for unreliable environments, none features a robustness verification mechanism. The solutions do not address integration of communication requirements and scenario-based connectivity descriptions into BPMN. Straight-forward mechanisms for modeling of communication-issue-related alternatives by domain experts are missing.

3 RESILIENT BPMN (*rBPMN*)

A good practice to design a BPMN extension was introduced by (Stroppi et al., 2011b) and (Braun et al., 2014). It starts with a requirements analysis of the


1. Analyse domain requirements
2. Perform equivalence check
3. Model CDME
4. Create BPMN+X
5. Create XML syntax
Focus of this paper
Created by model transformation rules from CDME
Created by model transformation rules from BPMN+X
Figure 1: Good practice of BPMN extension development, cf. (Stroppi et al., 2011b), (Braun et al., 2014).

target domain and an equivalence check to compare the requirements with existing BPMN concepts. Both steps help to identify missing parts of BPMN for unreliable environments. Following steps include modeling of the Conceptual Domain Model of the Extension (CDME) to address the missing parts, modeling the extended BPMN meta model (BPMN+X) to remain valid with BPMN and creation of the concrete syntax of XML schemas and files (5 steps shown in Figure 1).

This approach is also used for the development of *rBPMN* in this paper. Focus is on domain requirements, equivalence check and CDME, the creation of BPMN+X and XML syntax using transformation rules is described extensively in (Stroppi et al., 2011b). The following subsections will present *rBPMN* and its robustness verification mechanism.

3.1 Step 1: Domain requirements

Subsequent paragraphs will identify the requirements for robust process modeling. Agriculture will serve as an example for unreliable communication environments.

In agriculture, processes to till and harvest fields need to be defined, organized, controlled, monitored, adapted and documented. Many processes require collaboration of different actors (e.g. farmers, agricultural contractors, supporting machines, digital service providers, data hubs and authorities). While collaboration is required, modeling shall be time-efficient and respect trade secrets. Thus, the derived requirements are:

Req. 1: Ability to model collaborative agricultural processes including different actors (e.g. humans, machinery).

Req. 2: Ability to split collaborative agricultural work into reusable subsegments and to model them as black boxes.

The work needs to be controlled and monitored. Automatic and manual adaptions may be required, e.g. for automatically adapting machine parameters or manually deciding on machine failure replacements. Derived requirements:

Req. 3: Ability to adapt processes automatically based on predefined variables / events.

Req. 4: Ability to manually adapt running processes based on process state / proposed solutions.

Agriculture happens in rural, sparsely populated environments. The areas often lack cellular coverage, communication is delayed, intermittent or broken. Quality of actors' connectivity may change rapidly. Derived requirements:

Req. 5: Ability to define QoS requirements for communication and actors' connectivity characteristics.

Req. 6: Ability to ensure robustness even during intermittent or broken communication between actors.

Regardless of the challenging communication environment, processes shall operate optimally within given limits. Process robustness shall be verifiable before runtime. Derived requirements:

Req. 7: Ability to explicitly model optimal process operation at design time.

Req. 8: Ability to identify optimal process operation in a dynamically changing scenario at runtime.

Req. 9: Ability to verify process robustness for a given scenario at design time.

Finally, the extension shall follow the BPMN extension mechanism to be BPMN compliant. Derived requirement:

Req. 10: The extension shall comply with the BPMN meta model and semantics.

3.2 Step 2: Equivalence check

An equivalence check was performed to compare existing BPMN concepts with the requirements to model executable process diagrams for unreliable communication environments. The support level (SL) to fulfill the requirements is indicated by + ⇒ full support, o ⇒ limited support and - ⇒ no support in Table 1.

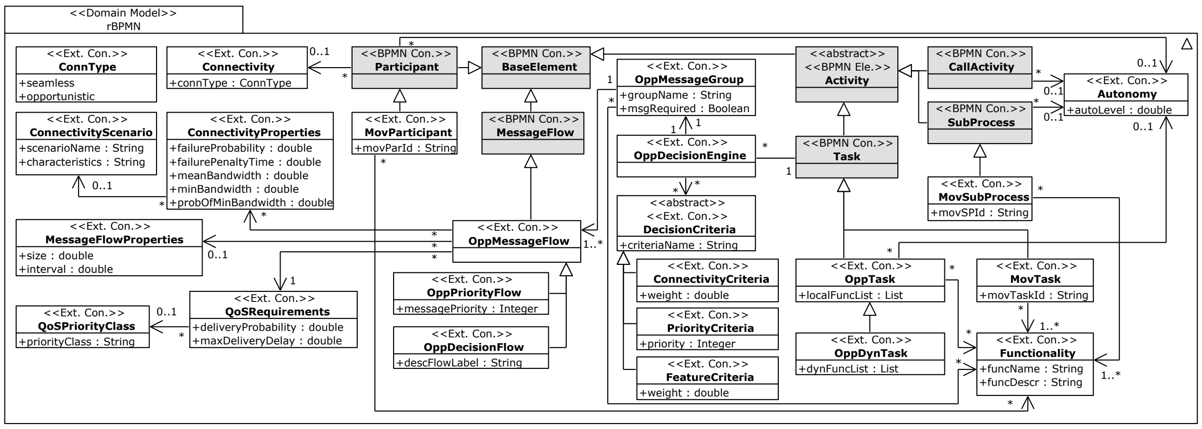
The equivalence check identifies BPMN concepts for the requirements 1, 2, 3 and 4. However, it also demonstrates the need for extension concepts to fulfill the requirements 5, 6, 7, 8 and 9 for communication, dynamics and decision modeling.

Application of the Decision Model and Notation (DMN) of OMG was evaluated and discarded. In DMN, a decision is taken using inputs and a set of rules structured by a decision table, resulting in an output (OMG, 2019). However, deciding on alternatives for broken communication requires to compare

Table 1: Equivalence check with BPMN concepts and their support-level (SL) for the identified domain requirements.

Req.	**Concept**	**Semantics** (and support declaration, where applicable)	**SL**
		General process modeling	
1, 2	Activity / Task	Part / step of a process.	+
1, 2	Process	Reusable container for a set / flow of chosen activities.	+
1, 2	Sub-process Call activity	Encapsulates / hides activities in processes, allows hierarchy levels, allows reuse.	+
1	Sequence flow	Coordinates the process flow.	+
3	Gateway Business rule task	Allows autonomous process flow decisions based on defined variables. Limited support: not designed for handling dynamics based on unreliable environments.	o
4	User task Manual task Ad-hoc sub-proc.	Allows user-based decisions for non-automated situations.	+
3, 4	Event Signal Conditional sequence flow Timer	Allows to react (dynamically) on messages, events, conditions, timings. Allows to dynamically create process instances / configurations.	+
1, 2	Text annotation	Specifies descriptive information, no influence on sequence flow.	+
		Collaboration modeling	
1	Participant Pool	Defines / structures different actors.	+
1, 2	Collapsed pool	Defines different actors, hides internals in black box.	+
1	Pool lanes	Separation of concerns / organization of activities within an actor.	+
		Communication modeling	
1	Message flow	Communication with other actors. Limited support: no possibility to specify alternative message flows / to specify flows as optional based on connectivity.	o
1	Message ItemDef.	Graphical element and name for a message. Limited support: frequency, size, relevance not included / not widely used in runable BPMN environments.	o
5, 7, 8, 9	Not available	No support: no BPMN concepts fulfill requirements 5, 7, 8, 9.	-
		Modeling of dynamics	
6	Not available	No support: no BPMN concept fulfills requirement 6.	-
		Modeling of decisions	
6, 7, 8	Decision table	Decision taking based on inputs, static rule sets in form of a decision table and outputs. Concept of Domain Model and Notation (DMN). No support: alternatives need to be compared to each other based on different criteria.	-

Support-Level-Declaration: + $\Rightarrow$ full support, o $\Rightarrow$ limited support, - $\Rightarrow$ no support



<<Domain Model>> rBPMN
<<Ext. Con.>> ConnType +seamless +opportunistic
<<Ext. Con.>> Connectivity +connType : ConnType
<<BPMN Con.>> Participant
<<BPMN Con.>> BaseElement
<<Ext. Con.>> OppMessageGroup +groupName : String +msgRequired : Boolean
<<abstract>> <<BPMN Ele.>> Activity
<<BPMN Con.>> CallActivity
<<Ext. Con.>> Autonomy +autoLevel : double
<<BPMN Con.>> SubProcess
<<Ext. Con.>> ConnectivityScenario +scenarioName : String +characteristics : String
<<Ext. Con.>> ConnectivityProperties +failureProbability : double +failurePenaltyTime : double +meanBandwidth : double +minBandwidth : double +probOfMinBandwidth : double
<<Ext. Con.>> MovParticipant +movParId : String
<<BPMN Con.>> MessageFlow
<<Ext. Con.>> OppDecisionEngine
<<BPMN Con.>> Task
<<Ext. Con.>> MovSubProcess +movSPId : String
<<abstract>> <<Ext. Con.>> DecisionCriteria +criteriaName : String
<<Ext. Con.>> MessageFlowProperties +size : double +interval : double
<<Ext. Con.>> OppMessageFlow
<<Ext. Con.>> ConnectivityCriteria +weight : double
<<Ext. Con.>> OppTask +localFuncList : List
<<Ext. Con.>> MovTask +movTaskId : String
<<Ext. Con.>> OppPriorityFlow +messagePriority : Integer
<<Ext. Con.>> PriorityCriteria +priority : Integer
<<Ext. Con.>> QoSPriorityClass +priorityClass : String
<<Ext. Con.>> QoSRequirements +deliveryProbability : double +maxDeliveryDelay : double
<<Ext. Con.>> OppDecisionFlow +descFlowLabel : String
<<Ext. Con.>> OppDynTask +dynFuncList : List
<<Ext. Con.>> Functionality +funcName : String +funcDescr : String
<<Ext. Con.>> FeatureCriteria +weight : double

Figure 2: Context Domain Model of the Extension (CDME) for *rBPMN*.

different options on the basis of defined criteria with each other.

So far, BPMN misses concepts to model QoS requirements for message flows, to flexible describe alternatives for failing message flows and to decide optimally on these alternatives during process runtime. In addition, it is not possible to verify process robustness for a given scenario at design time.

3.3 Step 3: Conceptual Domain Model of the Extension

The third step is represented by modeling the CDME. The core benefit of CDME development is to focus on the design of the modeling language for the applied domain without handling any restrictions set by the BPMN extension mechanism. The extension mechanism follows the principle of extension by addition rather than extension by generalization (OMG, 2011). However, generalization was used in the CDME to add extension concepts which will be explained subsequently. This will not result in any problems since transformation of CDME to BPMN+X in step 4 ensures BPMN validity.

After the missing parts for unreliable environments modeling have been identified by the domain requirements analysis and the equivalence check, the CDME shown in Figure 2 was designed. Figure 3 illustrates new graphical elements for the introduced opportunistic message flows, tasks and attributes.

The CDME adds new message flow types to the meta model to respect opportunistic communication characteristics. Based on existing BPMN message flows, OppMessageFlows enable the definition of QoS requirements, message flow properties and connectivity properties for specific scenarios. Using this information, a verification mechanism can compare required and provided connectivity to evaluate robust-

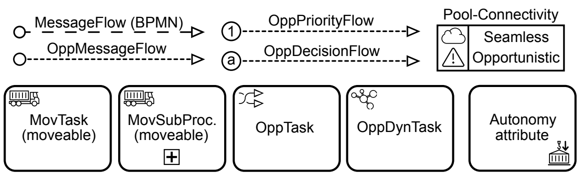


Figure 3: *rBPMN* message flows, extension tasks and attributes.

ness (cf. Section 3.4). Furthermore, priorities for deciding on alternative message flows in case of connectivity failures may be modeled using OppPriorityFlows. Priorities are recognizable by a number within the circle of a message flow.

More sophisticated decision taking on alternatives is possible using OppDecisionFlows. OppDecisionFlows annotate alternatives with alphabetic characters, displayed within the circle of the message flows. In combination with additional extension concepts for decision taking, different criteria can be weighted and combined in a matrix, allowing a decision engine to choose the optimal alternative in a flexible and dynamic manner. If necessary, other approaches for decision taking may be integrated into *rBPMN*.

Opportunistic messaging and traditional BPMN message flows may be used in a single diagram, especially helpful for scenarios with reliable communication segments.

A small number of new task types has been introduced. Moveable and opportunistic tasks represent the basis to locally move functionality between actors/system parts: MovTasks and MoveSubProcesses offer functionality that may be moved to other actors/system parts. OppTasks allow to locally store, execute and to integrate moved functionality into processes. OppDynTasks add more flexibility by the dynamic identification of alternatives that have not been explicitly modeled. Function descriptions help

Table 2: Extension concepts modeled to fulfill the requirements of unreliable communication environments.

Req.	Concept	Semantics
		Collaboration modeling
6	MovPraticipant	Participant offers moveable functionality to other actors / system parts.
		Communication modeling
5	OppMessageFlow	Possibly intermittent or broken communication with other actors / system parts. Allowed to be used with existing BPMN activities / tasks / participants.
7	OppPriorityFlow	Opportunistic message flow with explicitly defined priorities for alternatives modeling. A number within the message flow circle states the priority.
7, 8	OppDecisionFlow	Opportunistic message flow with implicit, criteria-based decision taking for alternatives. An alphabetic character within the message flow circle states the decision group.
9	MessageFlow-Properties	Describes message properties (e.g. frequency, size, relevance).
9	QoSRequirements	Defines QoS requirements for a message flow.
9	QoSPriorityClass	Defines a QoS hierarchy, to be used by QoSRequirements.
9	Connectivity	Defines a type of connectivity (seamless, opportunistic) for a participant.
9	Connectivity-Properties	Describes connectivity at the time of a message flow.
9	Connectivity-Scenario	Allows to group ConnectivityProperties to different scenarios.
		Modeling of dynamics
6	Autonomy	Allows to define a level of autonomy in case of broken connectivity for BPMN tasks (e.g. 4 OppMessageFlows, 3 with local functionality $\Rightarrow$ autonomy level of 75 %).
6	Functionality	Labels functionality, used to describe moveable process parts and to identify dynamic alternatives at runtime.
6	MovTask	Defines tasks that allow to move functionality to other actors / system parts.
6	MovSubProcess	Defines sub-processes that allow to move functionality to other actors / system parts.
6	OppTask	Task that is capable of accepting / operating local functionality.
6	OppDynTask	An OppTask that dynamically identifies alternatives (using functionality descriptions) which have not been modeled explicitly at design time.
		Modeling of decisions
7, 8	OppMessageGroup	Group of OppMessageFlows, where a decision on calling actors / system parts is needed.
6, 7, 8	OppDecisionEngine	Engine to choose OppMessageFlows based on decision criteria and engine parameters.
6, 7, 8	DecisionCriteria	Decision criteria used by decision engine. Three criteria have been defined: ConnectivityDecision, PiorityDecision, FeatureDecision.

to evaluate appearing actors/system parts as possible alternatives.

Visual recognition of locally moved functionality is supported by a task autonomy attribute shown by Figure 3. The connectivity type of actors and system parts is identifiable by pool attributes for seamless and opportunistic communication to the cloud/Internet.

Design of the CDME focuses on a lightweight, but powerful set of extension concepts. *rBPMN* may be combined with other extensions. Usage of BPMN elements with low practical usage is avoided [e.g. message / item def. in BPMN runtime engines such as (Camunda, 2019), more details in (Geiger et al., 2018)]. Additional remarks about the extension concepts and their semantics are summarized in Table 2.

Using the CDME and a set of transformation rules, BPMN+X (step 4) as well as the XML schemas and files (step 5) may be created. Due to space limitations, these steps are omitted in this paper and the reader is referred to transform illustration in (Stroppi et al., 2011b).

3.4 Robustness verification

Stable and failure-free execution of business processes is of high importance for most business domains. *rBPMN* prevents failing process executions by verifying the process robustness at design time. This section introduces *rBPMN's* verification mechanisms for a worst and average case robustness calculation in unreliable communication environments.

Equ. 1 determines the number of data packet frames N_f required to send a message ($\Rightarrow$ packet fragmentation). The message size M_s is divided by the payload size of a frame F_{pl}. The result is rounded up to the nearest whole number.

$$N_f = \left\lceil \frac{M_s}{F_{pl}} \right\rceil \tag{1}$$

Transferring messages requires protocols to organize and control communication. Protocol overhead of a data packet frame is represented by F_h and includes all protocol headers used within the frame. Since overhead created by the protocol stack can be several times higher than small amounts of application payload data, it is important to include F_h in the following calculations. Summarizing F_h and F_{pl} results in the total size of a data packet frame F.

In addition to M_s, N_h and F_h, a minimum bandwidth $BW_{min} > 0$ and a maximum delivery delay T_d should be provided to calculate a worst case robustness. Equ. 2 determines the worst case message transfer time T_{wc}. The message transfer is robust for $T_{wc} \leq T_d$, potentially not robust for $T_{wc} > T_d$ and certainly not robust for $BW_{min} = 0$.

$$T_{wc} = \frac{M_s + N_f * F_h}{BW_{min}} \tag{2}$$

Equ. 3 and 4 provide a robustness calculation more likely for many scenarios. The bandwidth BW is identified by BW_{min}, an average bandwidth BW_{av} and the probability for BW_{min} ($P_{BWmin} \in [0,1]$).

$$BW = BW_{min} * P_{BWmin} + BW_{av} * (1 - P_{BWmin}) \tag{3}$$

Subsequently, a failure probability $P_f \in [0,1]$ and a failure penalty time T_f are included in Equ. 4. The resulting value T_{ac} represents the message transfer time for an average case.

$$T_{ac} = \frac{M_s + N_f * F_h}{BW} + P_f * T_f \tag{4}$$

For recurring message transfers, Equ. 5 determines the number of messages N_m that fit into the requested message interval T_i. Afterwards, N_m is compared with the requested delivery probability $P_d \in [0,1]$. The recurring transfers are robust for $N_m \geq P_d$ and potentially not robust for $N_m < P_d$.

$$N_m = \frac{T_i}{T_{wc/ac}} \tag{5}$$

Domain experts may have difficulties estimating the required parameters for robustness calculations. Provisioning of representative bandwidth values for poor and average connectivity in the business domain helps to improve calculations. Guidelines for typical protocol stacks with their frame header and payload size can be provided (e.g. for TCP, IP, WiFi 802.11). Tools for monitoring and evaluating communication characteristics at process runtime may identify more sophisticated parameter values for future robustness optimizations.

4 USE CASE: A SLURRY APPLICATION

This section illustrates using *rBPMN* in an agricultural slurry scenario featuring different actors. Slurry is applied onto predefined subsections on a field based on its ingredients in the sense of precision farming. This approach ensures to comply with official regulations (e.g. distance to streams, protected areas).

A central process management (MGMT) located in the cloud controls the slurry application and assigns a task to a slurry spreader (SP). SP drives to the field

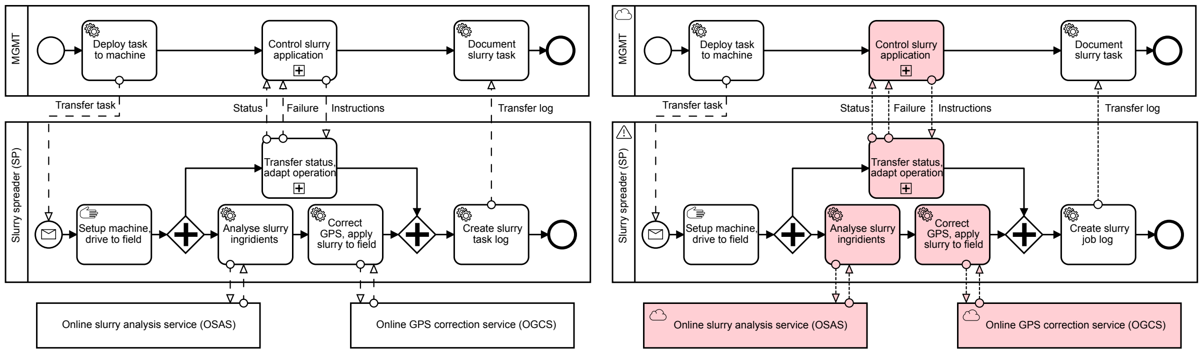

MGMT
Deploy task to machine
Control slurry application
Document slurry task
Transfer task
Status
Failure
Instructions
Transfer log
Slurry spreader (SP)
Transfer status, adapt operation
Setup machine, drive to field
Analyse slurry ingridients
Correct GPS, apply slurry to field
Create slurry task log
Create slurry job log
Online slurry analysis service (OSAS)
Online GPS correction service (OGCS)

Figure 4: Slurry scenario (a) in plain BPMN and (b) with OppMessageFlows and robustness verification.

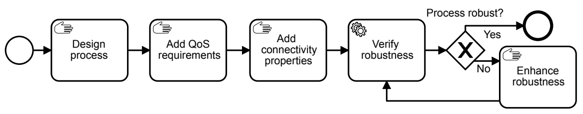


Figure 5: Adding robustness to processes with *rBPMN*.

Table 3: QoS and connectivity settings for messages.

QoS req.	Status	Fail./Instr.	Analysis	GPS corr.	Log	LGPS
P_d	0.5	1	1	0.9	1	0.9
T_d	10m	30s	5s	2s	1d	2s
Message properties						
M_s (KB)	500	1000	250	10	5000	10
T_i (s)	60	-	-	5	-	5
Connectivity properties						
P_f	0.3	0.3	0.3	0.3	0.3	0.1
T_f (s)	10	10	10	10	10	5
BW_{min} (kbps)	10	10	10	10	10	100
BW_{av} (kbps)	500	500	500	500	2500	1000
P_{BWmin}	0.5	0.5	0.5	0.5	0.3	0.2

Declaration: - $\Rightarrow$ not applicable

and starts the application of slurry with the help of an online slurry analysis service (OSAS) and an online GPS correction service (OGCS). During slurry application, SP continuously reports its status to MGMT and may receive instructions back. This is certainly true if SP is facing operational issues and sends a failure message to MGMT. Lastly, SP transfers a task log back to MGMT after finishing the slurry application. Figure 4 (a) displays the process in plain BPMN. The process would fail badly in many real-world-scenarios due to connectivity issues. However, a user would recognize the failing robustness not before the process is being executed.

Figure 4 (b) shows the slurry process using *rBPMN* after robustness verification. Red/gray parts indicate robustness issues, pointing out a non-stable slurry application process. Unreliable communication is recognizable by tightly dashed lines compared to non-problematic, standard BPMN message flows (cf. Figure 3). Pool attributes show the connectivity type of actors. Some actors have a seamless connection to the Internet (cloud sign), others may face interruptions or no Internet connectivity (warning sign).

This result was achieved by adding QoS and expected connectivity parameters for opportunistic message flows as described by Figure 5. QoS and connectivity parameters shown in Table 3 have been defined as an example for this paper. It is possible to model varying protocol overhead for different message transfers in *rBPMN*. Typical values for a combination of TCP, IP and WiFi 802.11 have been set in this example, resulting in a frame header size F_h of 82 byte and a frame payload size F_{pl} of 2230 byte (Kliazovich and Granelli, 2008).

The agricultural domain expert excluded the initial task transfer from MGMT to SP, because SP is located at a farm with solid connectivity at the time of transfer. Thus, the model in Figure 4 (b) features reliable and unreliable communication segments.

A worst case robustness verification identified several connectivity-related issues in Figure 4 (b), indicated by the red/gray color of tasks and message flows. While missing up to 50 percent of status reports from SP to MGMT is acceptable (P_d in Table 3), transfer of crucial failure information and instruction messages would fail. In addition, the actual slurry application could not operate: functionality calls to OSAS and OGCS would fail. At this point, *rBPMN* is able to enhance process robustness by

E_1: adding alternatives for message flows explicitly at design time, by

E_2: finding appropriate alternatives dynamically at runtime and by

E_3: moving (limited) functionality to SP.

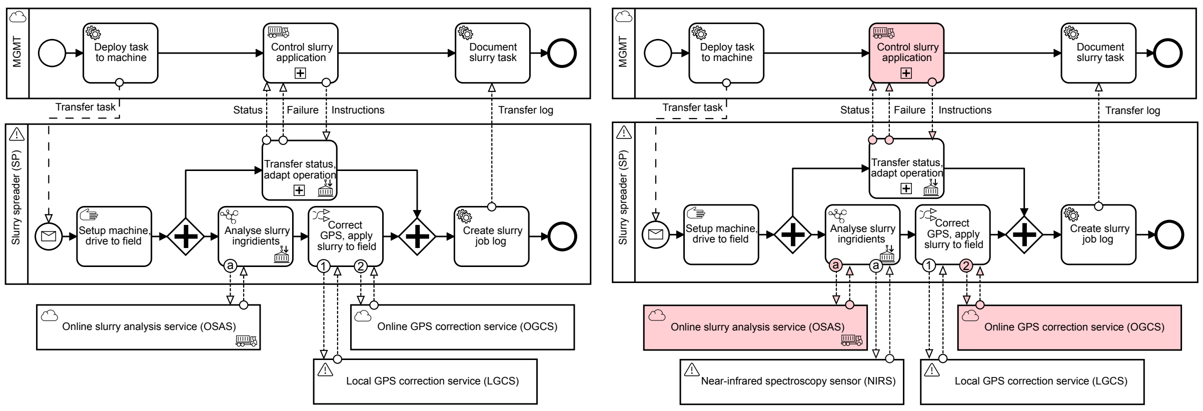

MGMT
Deploy task to machine
Control slurry application
Document slurry task
Transfer task
Status
Failure
Instructions
Transfer log
Slurry spreader (SP)
Transfer status, adapt operation
Setup machine, drive to field
Analyse slurry ingridients
Correct GPS, apply slurry to field
Create slurry job log
a
1
2
Online slurry analysis service (OSAS)
Online GPS correction service (OGCS)
Local GPS correction service (LGCS)
Near-infrared spectroscopy sensor (NIRS)

Figure 6: Slurry scenario (a) with robustness optimizations and (b) after final robustness verification.

All three robustness enhancement options have been used in Figure 6 (a):

Concerning E_1: A local GPS station at the field border (LGCS) was added to emit correction signals. Connectivity of LGCS to SP is solid during slurry application (cf. Table 3) due to its close proximity to SP.

Concerning E_2: Dynamic alternatives for the slurry ingredients analysis of SP have been included implicitly by choosing an OppDynTask. Since dynamic alternatives appear at runtime, they are not found in the graphical process model at design time. However, they appear in the decision matrix for the example slurry application illustrated in Table 4. Data provided by an ingredients laboratory (LAB) or a near-infrared spectroscopy sensor (NIRS) may be used, if they are present. Decision matrix explanations follow subsequently.

Concerning E_3: An autonomous control unit of MGMT and a limited calculation module of OSAS (OSAS-LF) have been designed as moveable functionality, illustrated by a moveable sub-process and a moveable black-box pool encapsulating functionality. The autonomy attributes on appropirate SP tasks indicate that functionality was moved locally.

Alternatives come with the challenge of decision taking. Priorities may be modeled explicitly with OppPriorityFlows, implicitly and more dynamically with OppDecisionFlows. In this example, the domain expert decided to use an OppTask with explicit OppPriorityFlows to get the GPS signal correction: first choice is LGCS, second is OGCS.

In contrast, the expert used an OppDynTask with OppDecisionFlows for the slurry ingredients analysis. Here, not all alternatives are graphically visible in the model. A decision matrix with different criteria (cf. Table 4) is used to define and dynamically decide for the optimal alternative at runtime. Alternatives include LAB, NIRS, OSAS and OSAS-LF. For this example, the decision engine is configured to choose the alternative with highest accuracy while having connectivity. More complex decision matrices and rules may be created, if required. For instance, the decision could also include feature properties such as an error ratio or a calculation time, connectivity aspects could include a connectivity prognosis as displayed by Table 4.

Table 4: Decision matrix using features and connectivity.

Feature	LAB	NIRS	OSAS	OSAS-LF
Accuracy (%)	90	75	65	55
Error ratio (%)	1	10	15	15
Calculation time delay (ms)	-	10	2000	-
Local performance cost (class)	-	-	-	1
Cost of usage (EUR)	25	10	2	5
Connectivity aspect				
Connectivity check	x	✓	(✓)	✓
Connectivity prognosis	x	✓	x	✓
Cancellation timeout (s)	0	1	5	0
Locally moved functionality	-	-	-	✓

Declaration: - ⇒ not applicable, x ⇒ failed, (✓) ⇒ limited success, ✓⇒ successful

Figure 6 (b) presents the final slurry robustness verification. Communication with MGMT, OSAS and OGCS is still unreliable. However, the slurry process is supposed to run stable and under optimal configuration: without having laboratory data on-hand, the ingredients alternatives decision is made for the dynamically appearing NIRS. It is rated as best available alternative by the decision engine (cf. Table 4). The backup alternative is the local functionality pro-

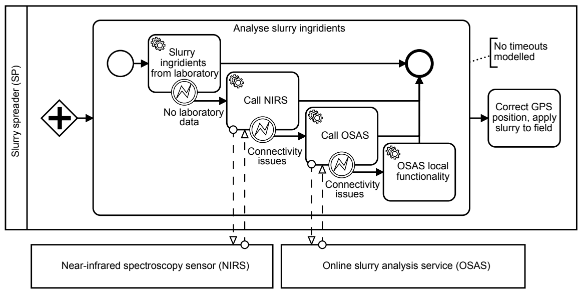

Slurry spreader (SP)
Analyse slurry ingridients
Slurry ingridients from laboratory
No laboratory data
Call NIRS
Connectivity issues
Call OSAS
Connectivity issues
OSAS local functionality
No timeouts modelled
Correct GPS position, apply slurry to field
Near-infrared spectroscopy sensor (NIRS)
Online slurry analysis service (OSAS)

Figure 7: Robustness modeling with BPMN error events (BPMN-EV).

vided by OSAS. MGMT might receive some status reports initiated by SP. Since the required QoS is not guaranteed for status, failure and instruction messages, local functionality of MGMT is picking up on SP in case of connectivity failures. This illustrates an important mechanism: communication attempts with other actors are possible even in rapidly changing network environments - after a defined timeout, local components take over operation. An optimal process operation is guaranteed.

5 ROBUST PROCESS MODELING IN BPMN 2.0

rBPMN represents a domain specific modeling language for unreliable communication environments. Mechanisms such as modeling alternatives for unavailable message flows are not exclusive to the extension and can be modeled within plain BPMN 2.0. This section compares robust process modeling using *rBPMN* with three approaches in plain BPMN that have been modeled for this paper.

The comparison is based on modeling the slurry ingredients analysis with multiple alternatives. Decision taking in the *rBPMN* implementation is shown in Table 4 and includes the alternatives LAB, NIRS, OSAS and OSAS-LF. Configuration can be changed by decision engine rules, the decision matrix stays untouched. Again, the engine is configured to decide on best accuracy while having connectivity.

BPMN-EV presented in Figure 7 is the first plain BPMN solution and uses error events to model alternatives. Alternatives are handled in fixed priority, one after another, and got encapsulated within an additional sub-process. If an alternative is unavailable due to connectivity issues or no data, an error event is thrown and the next option is addressed. Although timeouts are not modeled, the solution already requires high modeling effort and complicates the expressiveness of the diagram and its functionality. Priority changes require additional modeling effort. Dynamics do not exist, calling multiple actors in parallel due to bad connectivity is not possible.

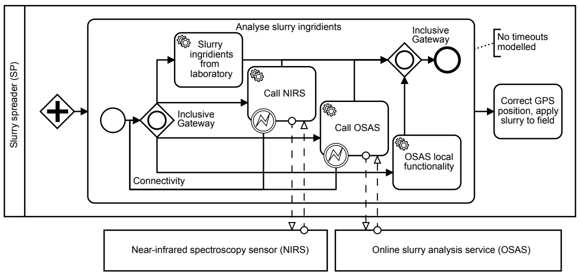

Slurry spreader (SP)
Analyse slurry ingridients
Inclusive Gateway
No timeouts modelled
Slurry ingridients from laboratory
Call NIRS
Inclusive Gateway
Call OSAS
OSAS local functionality
Connectivity
Correct GPS position, apply slurry to field
Near-infrared spectroscopy sensor (NIRS)
Online slurry analysis service (OSAS)

Figure 8: Robustness modeling with BPMN inclusive gateways (BPMN-GW).

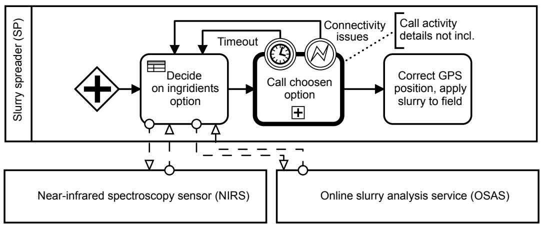

Slurry spreader (SP)
Timeout
Connectivity issues
Call activity details not incl.
Decide on ingridients option
Call choosen option
Correct GPS position, apply slurry to field
Near-infrared spectroscopy sensor (NIRS)
Online slurry analysis service (OSAS)

Figure 9: Robustness modeling with BPMN business rule tasks (BPMN-DMN).

Parallel communication is enabled by the second solution BPMN-GW displayed in Figure 8, again without timeouts for communication calls. Inclusive gateways add more flexibility compared to BPMN-EV. The expense is hidden within the inclusive gateway in which all rules need to be set up carefully to operate the process optimally. It is hardly possible to include all parameters for optimal process operation even with broadly defined gateway rules. Rule adaption is far from easy for agricultural domain experts. A sub-process is required to hide the modeling effort. Dynamically appearing alternatives are not captured.

The third solution BPMN-DMN in Figure 9 utilizes a business rule task and adds less graphical elements to the diagram. The business rule task is represented by a decision table of DMN, evaluating inputs using a set of rules to outputs. However, here we do not have a set of inputs that is evaluated against a static rule set. Instead, we want to compare dynamic alternatives with each other, choosing the most appropriate alternative to be invoked by a call activity. Hence, the solution is unable to provide the required flexibility. Furthermore, details of alternatives and their decision taking are hidden in the decision table.

Table 5 compares and evaluates the plain BPMN solutions with the introduced extension for unreliable communication environments. *rBPMN* is represented by a minimum set (*rBPMN-min*) implementing op-

Table 5: Comparison of robust modeling approaches.

Aspect	BPMN-EV	BPMN-GW	BPMN-DMN	*rBPMN-min*	*rBPMN-max*
Use of existing BPMN / DMN elements.	+	+	+	-	-
Graphical expressiveness of unreliable connectivity, alternatives and priorities.	o	o	-	+	+
Avoidance of additional graphical hierarchies for modeling of opportunistic characteristics.	o	o	o	+	+
Ability to integrate QoS requirements for opportunistic message flows between actors.	-	-	-	+	+
Ability to adapt process execution dynamically based on connectivity (run tasks exclusively / in parallel / locally moved functionality).	-	-	o	o	+
Ability to model & adapt to moveable / local / dynamically appearing functionality.	-	-	-	-	+
Domain experts: simplicity of robust modeling while focusing on problem domain.	-	-	-	+	+
Ability to verify process robustness for scenarios prior to runtime.	-	-	-	+	+
Summarized points	4	5	3	11	14

Declaration: + ⇒ full support / 2 points,
o ⇒ limited support / 1 point, - ⇒ no support / 0 points

portunistic message flows and connectivity parameters as well as a full implementation including priorities, decision taking and opportunistic tasks (*rBPMN-max*). All three plain BPMN solutions presented end up with a low number of points in the comparison. They are missing the required flexibility and do not verify robustness prior to process runtime. Error-prone and failing processes are the consequence.

6 CONCLUSION AND FURTHER RESEARCH

This paper introduces *rBPMN*, a valid BPMN extension for unreliable communication environments with limited, delayed, intermittent or broken connectivity. Using *rBPMN*, domain experts may model robust scenarios with dynamic alternatives for failing message transfers. Users may verify and enhance scenario robustness prior to runtime.

The BPMN meta model has been extended with opportunistic message flows, enabling the definition of QoS parameters and connectivity characteristics for robustness verification. Alternatives may be described explicitly with priorities or implicitly using decision criteria to be evaluated dynamically by decision engines. In case of no connectivity, locally moved functionality of other actors/system parts guarantees stable process operation. *rBPMN* has been designed as a lightweight, but powerful set of extension concepts. It is extendable by more complex parameters and criteria for decision taking and robustness verification, if required.

Further research objectives include practical evaluations of the proposed concepts for opportunistic messaging, moveable functionality and dynamic decision taking. In addition, research on a tool for assisted troubleshooting of robustness issues seems to be a promising help for domain experts.

ACKNOWLEDGEMENTS

The presented work is part of the OPeRAte project (OPeRAte, 2019). OPeRAte is supported by funds of the Federal Ministry of Food and Agriculture (BMEL) based on a decision of the Parliament of the Federal Republic of Germany via the Federal Office for Agriculture and Food (BLE) under the innovation support program.

REFERENCES

Betke, H. and Seifert, M. (2017). BPMN for disaster response processes. In *INFORMATIK 2017*, pages 1311–1324. Gesellschaft für Informatik, Bonn.

Bocciarelli, P. and D'Ambrogio, A. (2011). A BPMN extension for modeling non functional properties of business processes. In *Proceedings of the 2011 Symposium on Theory of Modeling & Simulation: DEVS Integrative M&S Symposium*, pages 160–168. Society for Computer Simulation International.

Bocciarelli, P., D'Ambrogio, A., Giglio, A., and Paglia, E. (2014). Simulation-based performance and reliability analysis of business processes. In *Proceedings of the 2014 Winter Simulation Conference*, pages 3012–3023. IEEE Press.

Bocciarelli, P., D'Ambrogio, A., Giglio, A., and Paglia, E. (2016). A BPMN extension to enable the explicit modeling of task resources. In *CIISE*, pages 40–47.

Bocciarelli, P., D'Ambrogio, A., Giglio, A., and Paglia, E. (2017). A BPMN extension for modeling cyber-physical-production-systems in the context of Industry 4.0. In *14th International Conference on Network-*

ing, Sensing and Control (ICNSC), pages 599–604. IEEE.

Bocciarelli, P. and DAmbrogio, A. (2014). A model-driven method for enacting the design-time qos analysis of business processes. *Software & Systems Modeling*, 13(2):573–598.

Braun, R. (2015). Behind the scenes of the BPMN extension mechanism principles, problems and options for improvement. In *3rd International Conference on Model-Driven Engineering and Software Development (MODELSWARD)*, pages 1–8. IEEE.

Braun, R. and Esswein, W. (2014a). Classification of domain-specific BPMN extensions. In *IFIP Working Conference on The Practice of Enterprise Modeling*, pages 42–57. Springer.

Braun, R. and Esswein, W. (2014b). Entwicklung einer BPMN-Extension für ressourcen-intensive Prozesse im Maschinenbau. *Tagungsband Multikonferenz Wirtschaftsinformatik*, 2014:1574–1586.

Braun, R., Schlieter, H., Burwitz, M., and Esswein, W. (2014). BPMN4CP: Design and implementation of a BPMN extension for clinical pathways. In *2014 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 9–16. IEEE.

Camunda (2019). Workflow and Decision Automation Platform, www.camunda.com (2019-09-03).

Domingos, D., Respício, A., and Martinho, R. (2016). Using resource reliability in BPMN processes. *Procedia Computer Science*, 100:1280–1288.

Dörndorfer, J. and Seel, C. (2017). A meta model based extension of BPMN 2.0 for mobile context sensitive business processes and applications. *13. Internationale Tagung Wirtschaftsinformatik (WI2017)*.

Dumas, M., La Rosa, M., Mendling, J., and Reijers, H. A. (2018). *Fundamentals of Business Process Management*. Springer, second edition.

Fall, K. (2003). A delay-tolerant network architecture for challenged internets. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 27–34. ACM.

Geiger, M., Harrer, S., Lenhard, J., and Wirtz, G. (2018). BPMN 2.0: The state of support and implementation. *Future Generation Computer Systems*, 80:250–262.

Graja, I., Kallel, S., Guermouche, N., and Kacem, A. H. (2016). BPMN4CPS: A BPMN extension for modeling cyber-physical systems. In *25th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, pages 152–157. IEEE.

ISO (2013). Internation Organization for Standardization: ISO:IEC 19510:2013, Information technology - Object Management Group Business Process Model and Notation.

Kliazovich, D. and Granelli, F. (2008). Packet concatenation at the ip level for performance enhancement in wireless local area networks. *Wireless Networks*, 14(4):519–529.

Kopp, O., Binz, T., Breitenbücher, U., and Leymann, F. (2012). BPMN4TOSCA: A domain-specific language to model management plans for composite applications. In *International Workshop on Business Process Modeling Notation*, pages 38–52. Springer.

Kozel, T. (2010). BPMN mobilisation. In *Proceedings of the European Conference of Systems: World Scientific and Engineering Academy and Society, WSEAS*.

Martinho, R. and Domingos, D. (2014). Quality of information and access cost of IoT resources in BPMN processes. *Procedia Technology*, 16:737–744.

Mazzola, L., Kapahnke, P., Waibel, P., Hochreiner, C., and Klusch, M. (2017). FCE4BPMN: On-demand qos-based optimised process model execution in the cloud. In *2017 International Conference on Engineering, Technology and Innovation (ICE/ITMC)*, pages 305–314. IEEE.

Meyer, S., Ruppen, A., and Hilty, L. (2015). The things of the internet of things in BPMN. In *International conference on advanced information systems engineering*, pages 285–297. Springer.

Meyer, S., Ruppen, A., and Magerkurth, C. (2013). Internet of things-aware process modeling: integrating IoT devices as business process resources. In *International conference on advanced information systems engineering*, pages 84–98. Springer.

OMG (2011). Object Management Group: Business Process Model and Notation (BPMN) 2.0 Specification, www.omg.org/spec/BPMN/2.0/About-BPMN (2019-09-03).

OMG (2014). Object Management Group: MetaObject Facility Specification, www.omg.org/mof (2019-09-03).

OMG (2019). Object Management Group: Decision Model and Notation (DMN) 1.2 Specification, www.omg.org/spec/DMN (2019-09-03).

OPeRAte (2019). Osnabrueck University of Applied Sciences: OPeRAte project, http://operate.edvsz.hs-osnabrueck.de (2019-09-03).

Pelusi, L., Passarella, A., and Conti, M. (2006). Opportunistic networking: data forwarding in disconnected mobile ad hoc networks. *IEEE Communications Magazine*, 44(11):134–141.

Respício, A. and Domingos, D. (2015). Reliability of BPMN business processes. *Procedia Computer Science*, 64:643–650.

Stroppi, L. J. R., Chiotti, O., and Villarreal, P. D. (2011a). A BPMN 2.0 extension to define the resource perspective of business process models. In *XIV Congreso Iberoamericano en Software Engineering*.

Stroppi, L. J. R., Chiotti, O., and Villarreal, P. D. (2011b). Extending BPMN 2.0: method and tool support. In *International Workshop on Business Process Modeling Notation*, pages 59–73. Springer.

Yousfi, A., Bauer, C., Saidi, R., and Dey, A. K. (2016). uBPMN: A BPMN extension for modeling ubiquitous business processes. *Information and Software Technology*, 74:55–68.