Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

uBPMN: A BPMN extension for modeling ubiquitous business processes

Alaaeddine Yousfi[a,b,*], Christine Bauer[c], Rajaa Saidi[a,d], Anind K. Dey[b]

[a] LRIT, Research Unit Associated to the CNRST (URAC 29), FSR, Mohammed V University, Rabat, Morocco
[b] Human-Computer Interaction Institute, Carnegie Mellon University, Pittsburgh, PA, USA
[c] Department of Information Systems and Information Management, University of Cologne, Cologne, Germany
[d] INSEA, Rabat, Morocco

ARTICLE INFO

Article history:
Received 25 July 2015
Revised 10 February 2016
Accepted 11 February 2016
Available online 23 February 2016

Keywords:
Business process modeling
Ubiquitous business process
Ubiquitous computing
uBPMN

ABSTRACT

Context: Business Process Model and Notation (BPMN) is the de facto standard for business process modeling. It was developed by the Object Management Group with support of the major organizations in the fields of software engineering and information systems. Despite its wide use, when it comes to representing ubiquitous business processes, this business process modeling language is lacking.

Objective: To address BPMN's deficiency in representing ubiquitous business processes, we extend it and present uBPMN (or ubiquitous BPMN).

Method: First, we analyze the modeling requirements for representing ubiquitous business processes. Based on the requirements, we conservatively extend the Meta-Object Facility meta-model and the XML Schema Definition of BPMN as well as extend the notation. The extension, that we call uBPMN follows the same outline as set by the Object Management Group for BPMN.

Results: The proposed uBPMN not only allows for modeling ubiquitous business processes but also lays the groundwork for potentially deploying a variety of ubiquitous computing technologies. We illustrate all of uBPMN's capabilities and benefits with real-life examples.

Conclusion: uBPMN extends BPMN v2.0 with new capabilities to deal with ubiquitous computing technologies.

1. Introduction

Nearly a decade after its official introduction in May 2004, the Business Process Model and Notation (BPMN) gained the upper hand in Business Process Modeling [8], both in academia and business. As of June 2015, it was referenced in more than 24,000 scientific publications and 300 patents as listed in Google Scholar/Patents. When compared to other business process modeling languages, numerous studies (e.g., [41,48]) emphasize that BPMN may be considered as the de facto standard for business process modeling. Further still, it is supported by big names in the fields of information systems and software engineering (see Section 6.3 in [42]). Last but not least, its groundbreaking features motivated the appearance of many engines supporting it such as Activiti [47], jBPM [57] and Oracle BPM [22].

Since its first release as v1.0 in May 2004, BPMN underwent three updates. Each update was introduced to allow BPMN to represent new process characteristics that were not covered by the version that preceded. The changes are thoroughly described by the Object Management Group (OMG) in each new release (e.g., changes from v1.2 to v2.0 are available in [42, p. 479]). Now, since its latest release as BPMN v2.0 in January 2011, Business Process Management has evolved a lot [15,24]. Particularly, many new process characteristics emerged that BPMN v2.0 cannot represent. For instance:

- **Example 1:** *The highway toll can be paid on the fly using the RFID (Radio Frequency IDentification) tag on the car windshield without stopping at any toll plaza.* The problem here is that automatic identification and data capture of the RFID tag and content cannot be appropriately represented by BPMN v2.0.
- **Example 2:** *The taxi is assigned to the customer based on her/his current location.* In this business rule, a mechanism to collect

* Corresponding author at: Human-Computer Interaction Institute, Carnegie Mellon University, Pittsburgh, PA, USA. Tel.: +1 412 268 9378.

E-mail addresses: aeyousfi@cmu.edu (A. Yousfi), bauer@wim.uni-koeln.de (C. Bauer), r.saidi@insea.ac.ma (R. Saidi), anind@cs.cmu.edu (A.K. Dey).

http://dx.doi.org/10.1016/j.infsof.2016.02.002
0950-5849/© 2016 Elsevier B.V. All rights reserved.

and quantify the current location of the user and assign the most appropriate taxi to her/him cannot be represented by BPMN v2.0.

- **Example 3:** *After capturing a sample of the music playing, the song is identified and added to the customer's music order.* Here, the problem is that BPMN v2.0 cannot accurately describe audio collection and sampling.

BPMN, in its latest release, offers five types of core modeling elements; Flow Objects, Data Objects, Connecting Objects, Swimlanes and Artifacts [42]. When tackling the aforementioned examples using the notation, many challenges arise. As such, what Flow Objects, Data Objects, Connecting Objects, Swimlanes and Artifacts can be used to *accurately* cope with the three business rules from the examples? One may suggest using the existing core modeling elements of BPMN v2.0. For example, one idea would be to represent these new capabilities with Artifacts, such as Text Annotations (also core elements of BPMN v2.0). While this may appear to be a viable solution from a modeling and design point of view, severe problems arise when it comes to Verification and Validation (V&V) [64] of the process models, as Text Annotations cannot be validated. Data Objects were introduced to deal with static data (e.g., file, database). A Text Annotation to indicate that Data Objects can handle dynamic data (e.g., sensor data) would be meaningless to a V&V algorithm. The same argument applies for the Flow Objects. Still, there will be additional difficulties, when the goal is to reach the transformation stage [19,20]. BPMN v2.0 falls short at describing those business rules because it does not contain core modeling elements that can accurately depict them.

The aforementioned examples that BPMN v2.0 cannot represent are examples that are based on ubiquitous computing. Ubiquitous computing (frequently referred to as ubicomp) was coined by Mark Weiser around 1988 [66]. Ubicomp denotes the third era of modern computing where one person owns and operates multiple computers (1 person, *n* computers). The first and the second were respectively mainframe computing (*n* persons, 1 computer) and personal computing (1 person, 1 computer). While human-computer interactions are typically administered via keyboards and mice as well as display, printers, and speakers, ubicomp adds to these state of the art input technologies such as sensors (e.g., accelerometer, gyrometer, geo-locating sensors), cameras, and microphones [59]. Ubicomp leverages the fact that computers pervade our lives by proposing solutions that bridge the gap between virtual systems and the physical environment (i.e., Internet of Things – IoT [33]).

Ubicomp capabilities may provide the basis for solving many issues in business process management, particularly with respect to process improvement [69–71], compliance [49], and security [68]. When ubicomp elements are included in a business process, we use the term "ubiquitous business process". *A ubiquitous business process is a location-independent business process that turns its business environment into a source of data and/or a target of outcome with the least of human interventions* [69]. Ubicomp capabilities include, for instance, Automatic Identification and Data Capture (AIDC) [60] (e.g., location-tracking [36], activity-sensing [17]) (which helps to overcome media breaks [26]), context awareness [1], augmented reality [5], sustainability [29], and ambient intelligence [46] that can be included in business processes. In fact, many organizations have already adopted such ubicomp capabilities to cope with the changing business environments and to remain competitive. For instance, UPS[1] overcomes media breaks between the physical and the digital world through bar-code tags to update the status of packages transiting through its logistics system. Netflix[2] and YouTube[3] use context awareness to recommend the most popular videos in the user's location (i.e., one type of context). TryLive[4] proposes augmented reality solutions to allow its users to virtually try on apparel. Nest[5] thermostats support sustainability. Google Now[6] enables ambient intelligence. However, as BPMN falls behind in representing these business scenarios, the organizations have designed them on their own. Their designs are digressions from the standard, since details touching the process lifecycle such as conformance [54] and compliance [55] remain overlooked. No formal V&V and/or transformation initiative(s) can take place, because the required conditions are not fulfilled. The process personnel in a process-oriented organization are classified into five categories; process owner, process manager, process participant, process analyst and process engineer [16]. Note, process analysts and engineers have direct interaction with BPMN. So, imagine the organization (i.e., process owner) wants to deploy augmented reality in one of its processes. Here, the process analyst has to set the specification for the new process. The process engineer has to make the specification concrete. Then, the analyst should validate its conformance before deploying it for the process participant. In this case, there is higher risk of having more back-and-forth discussions between analysts and engineers because there is no standard medium for them to clearly understand each other. Still, imagine after a certain time, the organization pushes for an improvement of the process and the initial team of analyst/engineer has changed. It would be a challenge for the new one to take over. Even with the old team being unchanged, the situation can also be challenging. This is analogous to writing code and coming back to it after time has passed. Even with comments in the code, it will take some time for the initial programmer to remember the details of the code before being able to weigh in with improvements. This problem can even be exacerbated in the case of a new programmer. Ultimately, without a clear-cut/common specification, the accessibility and use of process diagrams will only exist for the process analysts who created them.

In the scientific literature, we can find several attempts to deploy ubicomp or one of its capacities in business processes. Jung et al. [32] make a proposition of service integration, while Giner et al. [31] take a model driven approach to harmonize the dynamism of business processes and the complexity of ubicomp. While these two approaches attempt to tackle ubicomp as a whole, others focus on specific parts of it. For instance, the authors of [13,14,21] focus specifically on context awareness. Aoumeur et al. [3] and Zhu et al. [72] go even more specific and focus on one aspect of context awareness which is location awareness. A proposition of "*Smart Business Processes*" by means of an RFID integration is discussed in [2].

Although the potential advantages of including ubicomp in business processes are discussed across all the foregoing references, the critical question of how to design ubiquitous business processes remains unanswered. The major obstacle is that BPMN v2.0 cannot represent the ubiquitous computing input technologies. Consequently, it seems paramount to extend it. The present paper presents such an extension that we term "ubiquitous BPMN" (or, in short, uBPMN).

The remainder of this paper is organized as follows: In Section 2, we summarize related work that extended BPMN inside and outside the scope of ubicomp. Building on that, we enumerate the steps needed to coherently extend the notation.

[1] http://www.ups.com/.

[2] https://www.netflix.com/.

[3] https://www.youtube.com/.

[4] http://www.trylive.com/.

[5] https://nest.com/.

[6] https://www.google.com/landing/now/.

Following these steps, Section 3 highlights the ubicomp requirements for business process modeling while Section 4 describes our proposal for extending BPMN v2.0 to fulfill those requirements. In Section 5, we walk through an illustrative example about modeling a ubiquitous business process using our extended notation. We discuss the overall impact of our contribution in Section 6 and conclude the paper in Section 7.

2. Related work on BPMN extensions

In this section, we cover prior extensions of BPMN inside and outside the scope of ubicomp. In fact, BPMN has been extended by numerous contributions to describe different process characteristics across distinct domains [11]. These include, but are not limited to, quality management [51,56], performance measurement [9,27], e-health [12], security [52], resources [61], process tailoring [44], internal control [58] and time [30].

With regard to extensions in the domain of ubicomp, Gao et al. [32] add sensors and smart device business functions to close the information gap between the physical world and the processes. In the same vein, authors of [63,65] extend BPMN to represent Wireless Sensor Networks (WSNs). However, these extensions place sensors in a separate Pool. A Pool, based on the BPMN specification, is representative of a process participant. From a design standpoint, sensing (quantifying a physical data) is not that different from running a service. The BPMN Service Task is placed where appropriate in the process model without having a dedicated Pool. Meyer et al. [40], on the other hand, go by the specification and arrange the sensors in a separate BPMN lane. Finally, Appel et al. [4] extend BPMN to process event streams from the IoT, but, their work does not address how the event streams are captured and how this requirement is fulfilled throughout the BPMN extension. Additionally, it alters the semantics of the BPMN Events without considering that some of them do carry data (i.e., Message, Escalation, Error, Signal and Multiple) [42, p. 235].

Building on the previous paragraph, it is important to point out that WSNs are not ubicomp, and ubicomp is not only WSNs. WSNs are a type of input technologies within the discipline of ubiquitous computing. Still, some of the extensions are conservative (extended but staying within the specification) while others are not (and actually require changing the specification). In the former group, there is a lack of mechanisms to represent ubicomp within the process flow. While in the latter, no cogent reason has been provided to support why digressions from the BPMN standard are necessary, which hinders the dissemination and acceptance of their aforementioned propositions. To sum up, we believe and will show in the upcoming sections that ubicomp extensions of BPMN can be introduced in a conservative way; a way that neither alters nor contradicts the semantics of BPMN.

Before tackling the extension, we examined the most cited references that extended BPMN. They are [4,9–12,27,30,32,37,39,40, 44,51,52,56,58,61–63,65]. Our analysis identified two major steps that these works have in common:

1. They *analyze the requirements* of the target domain.
2. They *introduce the extension*. For the extension,
   (a) they clearly describe *the goals* and
   (b) the *scope* of the extension;
   (c) they extend the *core structure* and
   (d) they extend the *notation*.

We follow the same procedure. The analysis of the requirements of the target domain will be given in Section 3. Section 4 will first describe goals and scope of the extension. Then we will introduce the extension of the structure in Section 4.1 and then the extension of the notation in Section 4.2.

3. Ubicomp requirements for business processes

Stories of ubiquitous computing successes abound. For instance, in the transportation domain, ubicomp has been used to facilitate parking management, control the temperature of perishable food on the road (e.g., [31]) and compute the ride fare in public transportation [2]. Yet, the adoption of ubicomp elements, although considered a "*promising technological path of innovation*" [25], has only started to emerge in rather technology-oriented industries (e.g., traditional manufacturing, distribution and retail industries) [28,43], but has mostly negligible impact on other industries (e.g., insurance) [6]. As business applications increasingly become more social, user-focused, personalized, cloud-based, and/or mobile, organizations need to address the "ubiquitous elements" in their business processes [43]. These have substantial direct as well as mediated effects on process change and, thus, require more adaptive business processes [25,45]. However, the ubiquitous element is not yet considered in standard process models, and thus many innovation (e.g., [6,25]) and optimization (e.g., [43,45]) opportunities may be missed.

Enabling the myriad of ubicomp features consists of deploying a variety of input technologies for capturing the various kinds of "dynamic" contexts in the business environment. While contemporary business process modeling considers static contexts (e.g., [7]), a dynamic context has not received wide consideration in business process modeling. In ubiquitous computing, business environments are constantly subjected to changes. Additionally, they are influenced by distinct factors. Temperature, speech, audio, motion, emotion, etc., are all part of the dynamic context in a ubiquitous business environment and all have the potential for impacting the process flow. However, as long as deploying ubicomp requires such features that cannot be represented in BPMN v2.0, it will remain more challenging for their great potential to be realized. For instance, *How can a business process be context-aware if it does not have a mechanism to collect and quantify context? And how can a business process offer augmented reality if does not have an image input stream?* To overcome this issue, it is essential to enrich BPMN v2.0 with new elements and specifications, semantics and enable it to represent ubicomp input technologies.

A BPMN extension for ubicomp would have many benefits:

- Enabling processes to have faster, more accurate and less expensive data-flow and control-flow [69].
- Improving the process personnel's experience [34] by offering more customized processes that better reflect their business needs.
- Increasing the dissemination of the notation among researchers and practitioners in ubicomp by bridging the gap between their latest achievements and the discipline of business process management, and vice versa.
- Becoming a common reference to anyone who wants to engage in a V&V and/or transformation initiative.

It is believed that the amount of data we currently generate in 10 min equals all the data that was generated from prehistoric times up until 2003. We are generating data when we swipe our credit cards, run a red light, talk to each other, workout, adjust the air conditioning, etc. Even when one is reading this paper, she/he is generating data (e.g., how fast does the person read, what is the person's reading pattern, what sections/paragraphs does she focus on). The majority of this data cannot be processed using simple keyboards and mice, such as the eye movement when reading this paper. On a daily basis, we talk, listen to people/machines, talk, look at people/things and they look at us (e.g., camera), produce facial expressions, exhibit feelings, etc. This information is often thrown away, although it has potentially great value. As Clive

Humby once said[7]: "*Data is the new oil*". Ubicomp is the only well-established field that can minimize this waste through its panoply of new input and data collection technologies. As a result, it is of great importance to inject these technologies within business processes and to include them in managing the workflow.

Interactions with ubiquitous computing systems can take place via a panoply of mediums [53] such as smart homes, smart watches, smartphones, laptops, smart signs [35], etc. They also impact a wide range of applications such as speech recognition engines in web browsers (e.g., Google Chrome), mail/messaging, calendars, etc. Users can, now, interact with systems via audio commands, gesture (e.g., Microsoft Kinect[8]), motion/emotion, etc. They can even communicate with them via a set of standard representations (i.e., barcodes) or even via proxies. To circumscribe all these features, we started from what was already established around 2009 [50,59,66] and expanded the set following the latest achievements in ubicomp. To date, media break, context-awareness, augmented reality, sustainability and ambient intelligence are the main ubicomp features. As such, we distinguish five types of input technologies that are essential for enabling them. They are:

1. **Sensors**: quantify the physical data in the business environment. Examples of these are CO_2, GPS, accelerometer (a classification of sensors is available in [50,67]). They appear in many places (e.g., banks, supermarkets) and machines (e.g., phones, cars) that we visit and interact with on a regular basis. At present, a typical smartphone can have around fifteen sensors.
2. **Smart readers**: read an information represented in a standardized fashion such as bar codes, RFID, NFC, biometrics and magnetic stripes. These play a major role in identifying objects. For instance, bar codes identify goods in supermarkets and packages for shipping.
3. **Cameras**: capture the information as an image (a video is a series of moving images with a minimum speed of sixteen frames per second). Images are fundamental for object recognition and augmented reality. For example, the image input technology is used to identify license plates, analyze facial expressions and virtually try on apparel.
4. **Microphones**: capture the information as an audio segment. Note, audio and video streams are independent from each other. For audio/video collection, both of them must be used. Microphones are used in situations such as firing up audio commands (e.g., Google Now, Amazon Echo[9], Apple Siri[10]) and identifying music (e.g., Shazam[11]).
5. **Collectors**: gather information from web-services, local or remote files (e.g., Dropbox) or databases (e.g., obtaining weather data from a web service) or even proxy devices (e.g., get GPS data from a different device than the user's because both devices share the same context [23]).

In what follows, we propose uBPMN as a viable solution to represent the aforementioned input technologies and lay the groundwork for the latest ubicomp features. We illustrate each addition with concrete examples from the business world and summarize all the details in an final example.

4. uBPMN

Goals: The main goal of this paper is to provide a conservative extension (extended by the notation) of BPMN that allows the creation of end-to-end ubiquitous business processes as well as the portability of their definitions. The creation stems from the fact that the new notation lays the groundwork for deploying many ubiquitous computing features such as AIDC, context awareness, augmented reality, sustainability and ambient intelligence. For the portability, the new extension can communicate a wide variety of ubiquitous computing information to a wide variety of audiences by creating a straightforward bridge for the gap between the process personnel; e.g., from process analysts who set up the initial drafts of the processes, to process engineers responsible for implementing the ubiquitous technology that will perform those processes. Out extension is introduced to provide them with a standard visualization mechanism to design, manage, control, monitor and communicate ubiquitous business processes in a smooth fashion.

Scope: To overcome the shortfalls of BPMN v2.0 with regard to ubicomp, we introduce ubiquitous BPMN (hereafter shortened to uBPMN). uBPMN is our conservative extension of BPMN that allows the creation of end-to-end ubiquitous business processes and guarantees their portability. Everything true about BPMN is also true about uBPMN.

The ubicomp extension of BPMN will form a clear conceptual link with ubiquitous business processes. It will also provide suitable modeling elements to describe ubicomp interactions within the process flow. The proposed extension advances BPMN by offering support for modeling ubiquitous business rules. These cannot be represented using traditional BPMN. Generally speaking, a *ubiquitous business rule* is a business rule that uses a ubiquitous computing technology to define or constrain the aspects of the workflow (e.g., scan the bar-code to update the package status, read the RFID tag to generate the invoice). For better comprehensibility, we explain these based on two examples:

- The general trend for small privately-owned businesses (e.g., convenience stores, barbers, bakeries) is: the customer walks in, waits for her/his turn, gets the product or the service achieved, pays for it and leaves. For example, when in a local barber shop; the customer walks in, waits for his turn, gets his hair done, pays cash and leaves. All the business rules in this scenario do not depict ubiquity. Consequently, the process can be designed using BPMN.
- All around the world, the status of each package transiting via the UPS logistics system is updated upon arrival and before departure, i.e., arrival scan and departure scan. Here, ubiquity can be depicted for two reasons related to the ubiquitous business process definition. First, the process is *location-independent* since the scan can take place anywhere in the world, e.g., UPS Worldport in Louisville, UPS truck. Second, UPS packages are smart objects distinguished by bar codes, which is a ubicomp input technology. Each package can report its state when scanned. Consequently, the process needs to be designed using uBPMN.

4.1. Structure

The BPMN core structure appears in two representations: A Meta-Object Facility (MOF) meta-model describing the concepts and an XML Schema Definition (XSD) setting the interchange format [18,62]. The ubicomp core structure extension targets both of these key pillars.

The MOF class diagram meta-model of BPMN is divided and presented in segments across the specification manual [42]. Here, we aggregate those segments to the most optimal expressive uBPMN diagram. Due to space allocation, we show the aggregation result in two figures; Figs. 1 and2. The classes in white belong to the BPMN v2.0 standard while the classes in gray reflect the uBPMN extension. Fig. 1 shows the extensions of "*Activity*" from

[7] Quoted from the ANA senior marketers summit.
[8] http://www.xbox.com/en-US/xbox-one/accessories/kinect-for-xbox-one.
[9] http://www.amazon.com/oc/echo/.
[10] https://www.apple.com/ios/siri/.
[11] shazam.com.

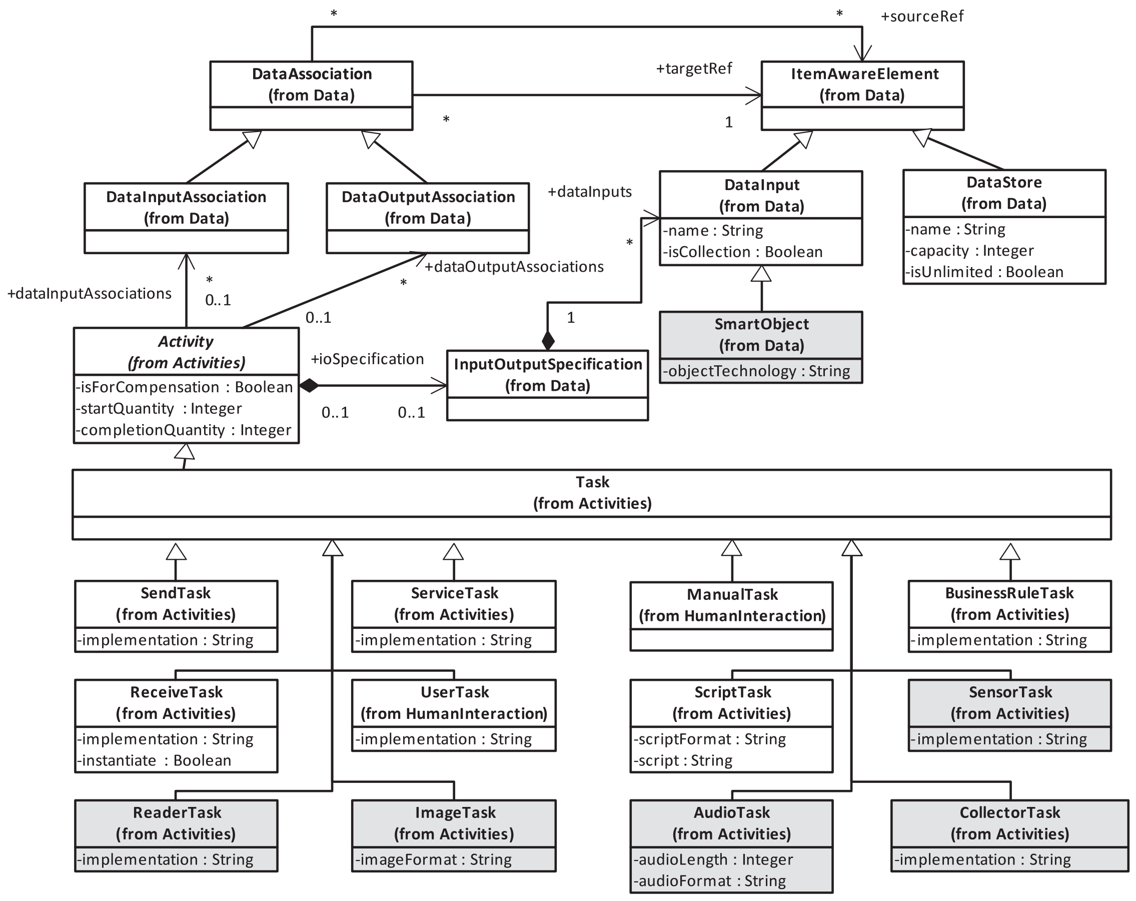
DataAssociation (from Data)
ItemAwareElement (from Data)
+sourceRef
+targetRef
DataInputAssociation (from Data)
DataOutputAssociation (from Data)
DataInput (from Data) -name : String -isCollection : Boolean
DataStore (from Data) -name : String -capacity : Integer -isUnlimited : Boolean
+dataInputs
+dataInputAssociations
+dataOutputAssociations
SmartObject (from Data) -objectTechnology : String
Activity (from Activities) -isForCompensation : Boolean -startQuantity : Integer -completionQuantity : Integer
+ioSpecification
InputOutputSpecification (from Data)
Task (from Activities)
SendTask (from Activities) -implementation : String
ServiceTask (from Activities) -implementation : String
ManualTask (from HumanInteraction)
BusinessRuleTask (from Activities) -implementation : String
ReceiveTask (from Activities) -implementation : String -instantiate : Boolean
UserTask (from HumanInteraction) -implementation : String
ScriptTask (from Activities) -scriptFormat : String -script : String
SensorTask (from Activities) -implementation : String
ReaderTask (from Activities) -implementation : String
ImageTask (from Activities) -imageFormat : String
AudioTask (from Activities) -audioLength : Integer -audioFormat : String
CollectorTask (from Activities) -implementation : String

Fig. 1. uBPMN meta-model (part I).

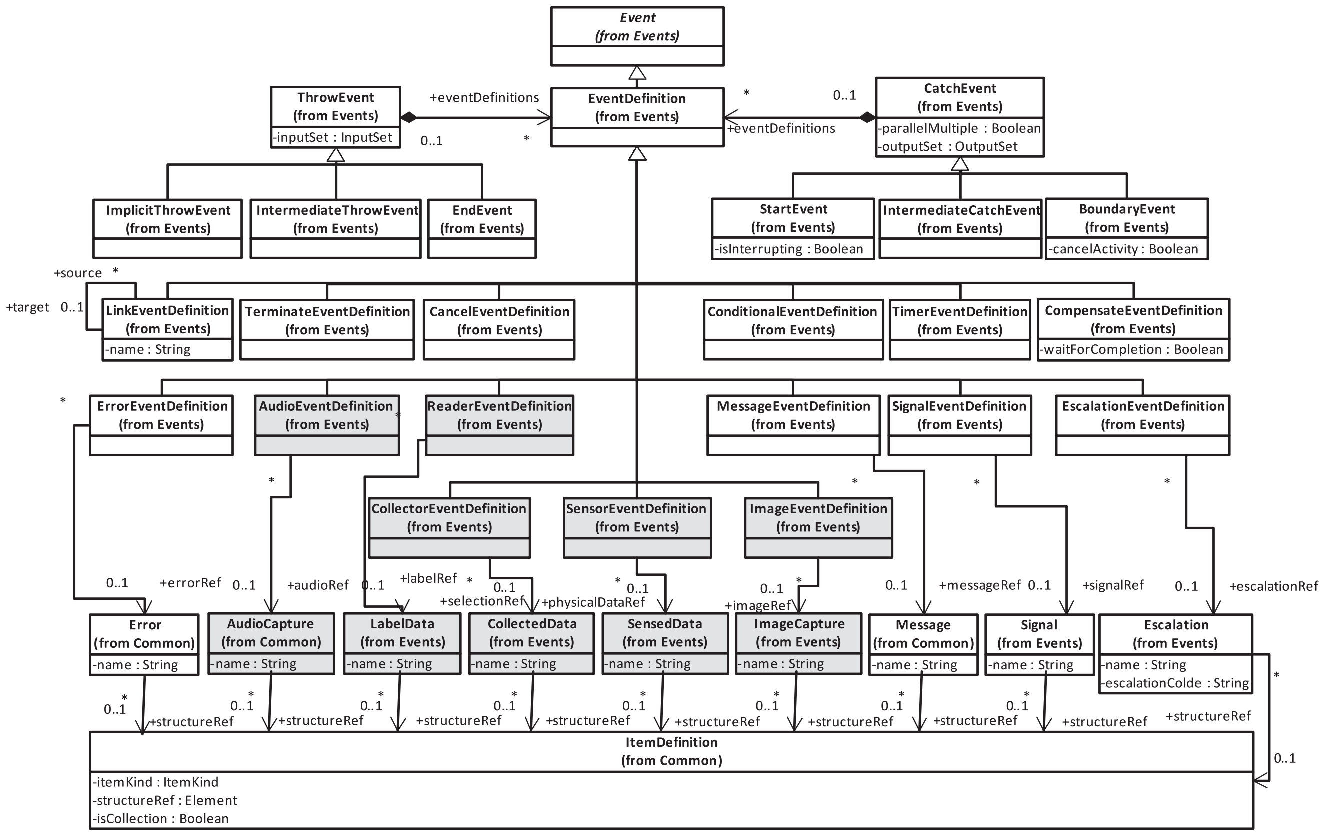
Event (from Events)
ThrowEvent (from Events) -inputSet : InputSet
+eventDefinitions
EventDefinition (from Events)
CatchEvent (from Events) -parallelMultiple : Boolean -outputSet : OutputSet
ImplicitThrowEvent (from Events)
IntermediateThrowEvent (from Events)
EndEvent (from Events)
StartEvent (from Events) -isInterrupting : Boolean
IntermediateCatchEvent (from Events)
BoundaryEvent (from Events) -cancelActivity : Boolean
+source
+target
LinkEventDefinition (from Events) -name : String
TerminateEventDefinition (from Events)
CancelEventDefinition (from Events)
ConditionalEventDefinition (from Events)
TimerEventDefinition (from Events)
CompensateEventDefinition (from Events) -waitForCompletion : Boolean
ErrorEventDefinition (from Events)
AudioEventDefinition (from Events)
ReaderEventDefinition (from Events)
MessageEventDefinition (from Events)
SignalEventDefinition (from Events)
EscalationEventDefinition (from Events)
CollectorEventDefinition (from Events)
SensorEventDefinition (from Events)
ImageEventDefinition (from Events)
+errorRef
+audioRef
+labelRef
+selectionRef
+physicalDataRef
+imageRef
+messageRef
+signalRef
+escalationRef
Error (from Common) -name : String
AudioCapture (from Common) -name : String
LabelData (from Events) -name : String
CollectedData (from Events) -name : String
SensedData (from Events) -name : String
ImageCapture (from Events) -name : String
Message (from Common) -name : String
Signal (from Events) -name : String
Escalation (from Events) -name : String -escalationColde : String
+structureRef
ItemDefinition (from Common) -itemKind : ItemKind -structureRef : Element -isCollection : Boolean

Fig. 2. uBPMN meta-model (part II).

the category *Flow Object* and "*Data Input*" from the category *Data* while Fig. 2 illustrates the extensions of "*Event*" from the category *Flow Object*.

Concerning Fig. 1, five new *Task* structures are introduced. The classes `SensorTask`, `ReaderTask`, `ImageTask`, `AudioTask` and `CollectorTask` inherit the attributes and model associations of `Activity`. The attribute `implementation` is of type String and refers to the sensing and/or reading and/or collecting technology implemented (depending on the class in which it appears). `imageFormat` is of type String and refers to the format of the image captured while `audioFormat` and `audioLength` (in seconds) are of type String and Integer, and refer to the format and length of the captured audio segment respectively. A new *Data* structure is also introduced. It is represented by the class `SmartObject` which inherits the attributes and model associations of `DataInput`. The additional attribute `objectTechnology` is a String describing the ubicomp input technology used.

As indicated in Fig. 2, we introduce five new *Event* structures for uBPMN. The classes `SensorEventDefinition`, `ReaderEventDefinition`, `ImageEventDefinition`, `AudioEventDefinition` and `CollectorEventDefinition` inherit the attributes and model associations of `BaseElement` through `EventDefinition`. Similar to the BPMN events Message, Escalation, Error, Signal and Multiple, all of the five uBPMN events carry data. Within the meta-model, this specification is expressed by connecting `SensedData`, `LabelData`, `ImageCapture`, `AudioCapture` and `CollectedData` to `ItemDefinition`. The attribute `name` in the new classes has the same semantic as the old ones.

The XSD specification of BPMN v2.0 is in five files: *BPMN20.xsd* being the main file and four namespaces (*BPMNDI.xsd, DC.xsd, DI.xsd* and *Semantic.xsd*). The uBPMN upgrade targets the last file since it holds the BPMN semantics. Due to its lengthy content (most lengthy amongst the others), we omit the BPMN v2.0 aspects and focus only on the extensions (see Listing 1[12]). The ellipsis refers to the existing code in the BPMN v2.0 specification. `QName` and `##unspecified` preserve their semantics [42, pp. 92–167]; the first references a definition existing in an external XML file while the second leaves the implementation technology window open because there exists numerous sensing/reading/collecting technologies and more are yet to come. Note, since we altered the original *Semantic.xsd* file, we modify the values of the attributes `xmlns` and `targetNamespace` to `ubpmn`, to indicate this change.

4.2. Notation

The BPMN notation offers a multitude of modeling elements that help shape the business process and define its behavior. The additional ubicomp elements will provide suitable modeling concepts for ubicomp (e.g., collecting GPS coordinates) and be adapted to incorporate new behavior within business processes (e.g., context awareness).

As discussed in the previous section, the extensions target "*Activity*" and "*Event*" in the category *Flow Object* as well as "*Data Input*" from the category *Data*. Consequently, this section follows up by extending the notation and introducing the equivalent modeling elements.

4.2.1. Items and data

BPMN v2.0 offers four types of data modeling elements; *Data Object* (including Data Object Reference and Data Object Collection), *Data Input, Data Output* and *Data Store* (including Data Store Reference). The Data Object/Store References are a way to reuse Data Objects/Stores in the same diagram. Unlike a Data Object whose life-cycle is tied to its parent Process/Sub-Process, a Data Store is about persistence of the data beyond the scope of the process. Data Inputs and Data Outputs respectively represent data requirements and data outcomes in a process.

Following the previous classification, uBPMN introduces a new data modeling element that we name *Smart Object*. Theoretically speaking, a smart object is a plain physical object enriched with a ubiquitous computing input technology to report its state. For example, a box is a plain physical object. On the other hand, a box with a descriptive bar-code tag becomes a smart object because the bar-code tag can report the state of the box with no media break. The *Smart Object* addition reflects the uBPMN extension for the category Data following the structure of the meta-model presented in Figs. 1 and 2, the XSD of Listing 1 and the guidelines explained below:

- ***Smart Object***
  A ***Smart Object*** is a *Data Input* object enriched with a ubicomp input technology to report its state with the least of human interventions. It denotes a declaration that a particular kind of data will be collected by either a Sensor, Smart Reader, Microphone or a Camera. The *Smart Object* has the same notation as the BPMN v2.0 *Data Object*. We place a brain logo in the upper left corner of its visual representation to indicate that it is a Smart Object. (See Fig. 3)

4.2.2. Activities

The OMG defines three types of activities in the BPMN v2.0 standard; *Task, Sub-Process* and *Call Activity*. A Task is an atomic activity that describes an action taken in the process flow (e.g., input student ID) while the Sub-Process is a self-contained composite subset of a process (e.g., input student ID, update student record, send a notification to the student; two Tasks and one Message Throw Event). The Call Activity allows the inclusion of reusable Tasks and Processes by invoking them from the Global Process. Activating the Call Activity results in transferring control to the Global Process (e.g., calling either the latter Task or Sub-Process of updating the student's records which figure in the Global Process).

Being a subtype of Activity, the element Task has also seven subtypes (i.e., Send, Receive, User, Service, Manual, Script and Business Rule). They are used to describe the process flow in a very specific fashion. Within uBPMN, we add five new types to the Task subset, for a total of twelve elements. The additional elements are Sensor, Reader, Image, Audio and Collector. They reflect our uBPMN extension for the Flow Object Activity following the structure of the meta-model shown in Figs. 1 and 2, the XSD of Listing 1 and the guidelines explained below:

- ***Sensor Task***
  A ***Sensor Task*** is a *Task* that uses some sort of sensor (e.g., [50,67]) to collect a particular type of information in the business environment. A Sensor Task object shares the same shape as the Task, which is a rectangle with rounded corners. However, there is a signal clip art in the upper left corner of the shape indicating that the Task is a Sensor Task (see Fig. 4). In addition, the ***Sensor Task*** has a maximum of one connection to a Smart Object. Note, in the case of GPS, one connection is enough.
  Example: The process participant is getting out of work and she requests a taxi. The taxi is assigned to participant after sensing her current location. Here, the current location (latitude and longitude) is sensed from the Smart Object GPS satellite and transferred to the taxi company system. The ubiquitous

[12] Built and validated with Liquid XML Studio 2014.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema elementFormDefault="qualified" attributeFormDefault="unqualified" xmlns="ubpmn"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema" targetNamespace="ubpmn">


...


<xsd:element name="sensorTask" type="tSensorTask" substitutionGroup="flowElement"/>
<xsd:complexType name="tSensorTask">
  <xsd:complexContent>
    <xsd:extension base="tTask">
      <xsd:attribute name="implementation" type="tImplementation" default="##unspecified"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="readerTask" type="tReaderTask" substitutionGroup="flowElement"/>
<xsd:complexType name="tReaderTask">
  <xsd:complexContent>
    <xsd:extension base="tTask">
      <xsd:attribute name="implementation" type="tImplementation" default="##unspecified"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="collectorTask" type="tCollectorTask" substitutionGroup="flowElement"/>
<xsd:complexType name="tCollectorTask">
  <xsd:complexContent>
    <xsd:extension base="tTask">
      <xsd:attribute name="implementation" type="tImplementation" default="##unspecified"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="imageTask" type="tImageTask" substitutionGroup="flowElement"/>
<xsd:complexType name="tImageTask">
  <xsd:complexContent>
    <xsd:extension base="tTask">
      <xsd:attribute name="imageFormat" type="xsd:string"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="audioTask" type="tAudioTask" substitutionGroup="flowElement"/>
<xsd:complexType name="tAudioTask">
  <xsd:complexContent>
    <xsd:extension base="tTask">
      <xsd:attribute name="audioLength" type="xsd:integer"/> <!-- audioLength (in seconds) -->
      <xsd:attribute name="audioFormat" type="xsd:string"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="smartObject" type="tSmartObject" />
<xsd:complexType name="tSmartObject">
  <xsd:complexContent>
    <xsd:extension base="tDataInput">
      <xsd:attribute name="objectTechnology" type="xsd:string"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="sensedData" type="tSensedData" substitutionGroup="rootElement"/>
<xsd:complexType name="tSensedData">
  <xsd:complexContent>
    <xsd:extension base="tRootElement">
      <xsd:attribute name="name" type="xsd:string"/>
      <xsd:attribute name="structureRef" type="xsd:QName"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="sensorEventDefinition" type="tSensorEventDefinition" substitutionGroup="eventDefinition"/>
<xsd:complexType name="tSensorEventDefinition">
  <xsd:complexContent>
    <xsd:extension base="tEventDefinition">
      <xsd:attribute name="physicalDataRef" type="xsd:QName"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="labelData" type="tLabelData" substitutionGroup="rootElement"/>
<xsd:complexType name="tLabelData">
```

Listing 1. uBPMN XSD.

```
   <xsd:complexContent>
      <xsd:extension base="tRootElement">
         <xsd:attribute name="name" type="xsd:string"/>
         <xsd:attribute name="structureRef" type="xsd:QName"/>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>

<xsd:element name="readerEventDefinition" type="tReaderEventDefinition" substitutionGroup="eventDefinition"/>
<xsd:complexType name="tReaderEventDefinition">
   <xsd:complexContent>
      <xsd:extension base="tEventDefinition">
         <xsd:attribute name="labelRef" type="xsd:QName"/>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>

<xsd:element name="collectedData" type="tCollectedData" substitutionGroup="rootElement"/>
<xsd:complexType name="tCollectedData">
   <xsd:complexContent>
      <xsd:extension base="tRootElement">
         <xsd:attribute name="name" type="xsd:string"/>
         <xsd:attribute name="structureRef" type="xsd:QName"/>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>

<xsd:element name="collectorEventDefinition" type="tCollectorEventDefinition" substitutionGroup="eventDefinition"/>
<xsd:complexType name="tCollectorEventDefinition">
   <xsd:complexContent>
      <xsd:extension base="tEventDefinition">
         <xsd:attribute name="selectionRef" type="xsd:QName"/>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>

<xsd:element name="audioCapture" type="tAudioCapture" substitutionGroup="rootElement"/>
<xsd:complexType name="tAudioCapture">
   <xsd:complexContent>
      <xsd:extension base="tRootElement">
         <xsd:attribute name="name" type="xsd:string"/>
         <xsd:attribute name="structureRef" type="xsd:QName"/>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>

<xsd:element name="audioEventDefinition" type="tAudioEventDefinition" substitutionGroup="eventDefinition"/>
<xsd:complexType name="tAudioEventDefinition">
   <xsd:complexContent>
      <xsd:extension base="tEventDefinition">
         <xsd:attribute name="audioRef" type="xsd:QName"/>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>

<xsd:element name="imageCapture" type="tImageCapture" substitutionGroup="rootElement"/>
<xsd:complexType name="tImageCapture">
   <xsd:complexContent>
      <xsd:extension base="tRootElement">
         <xsd:attribute name="name" type="xsd:string"/>
         <xsd:attribute name="structureRef" type="xsd:QName"/>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>

<xsd:element name="imageEventDefinition" type="tImageEventDefinition" substitutionGroup="eventDefinition"/>
<xsd:complexType name="tImageEventDefinition">
   <xsd:complexContent>
      <xsd:extension base="tEventDefinition">
         <xsd:attribute name="imageRef" type="xsd:QName"/>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
</xsd:schema>
```

Listing 1. Continued

business process fragment is shown in Fig. 5. Note, we disregard designing the communication with the taxi company system and focus solely on the Sensor Task segment. *The Link Intermediate Events of type Catch and Throw indicate that what is presented is only a fragment. Text Annotations can also be used to describe the sensing technology used. These comments are valid for all the examples that follow in this paper.*

- ***Reader Task***
  A ***Reader Task*** is a *Task* that uses some sort of ubiquitous reading technology (e.g., Bar-code, RFID, NFC, Biometrics, Magnetic

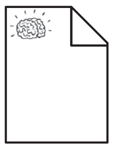

Fig. 3. Smart Object.

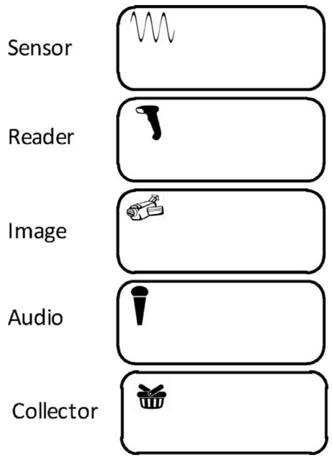

Fig. 4. uBPMN tasks.

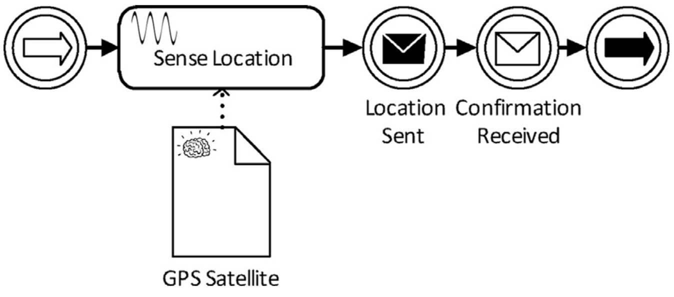

Fig. 5. Sensor Task example.

Stripes) to collect a particular information in the business environment. A Reader Task object shares the same shape as the Task, which is a rectangle with rounded corners. However, there is a scanner in the upper left corner of the shape to indicate that the Task is a Reader Task (see Fig. 4). In addition, the ***Reader Task*** has exactly one single connection to a Smart Object.
Example: The process participant is in a virtual subway store (e.g., Homeplus in South Korea[13]). He can add items to his cart by scanning their bar-codes. Here, the product is identified by reading its bar-code. The ubiquitous business process fragment is shown in Fig. 6.

- ***Image Task*** An ***Image Task*** is a *Task* that uses a video stream to collect a particular information in the business environment. The data collected is an image capture; one frame. For multiple frames (i.e., video), the Task can be looping. An Image Task object shares the same shape as the Task, which is a rectangle that has rounded corners. However, there is a camera in the upper left corner of the shape indicating that the Task is of type Image (see Fig. 4). In addition, the ***Image Task*** has a single connection to one Smart Object.
Example: The process participant walks by a street ad sign. She sees a product she likes but she does not have access to the product's identifier (e.g., bar-code). Numerous object recognition algorithms can now be used to upgrade the physical object (snapshot) to a Smart Object. The most famous among all is the Scale-Invariant Feature Transform (or SIFT) introduced by David Lowe [38]. The user can add the product to her cart by taking a picture of it. The ubiquitous business process fragment is shown in Fig. 7.

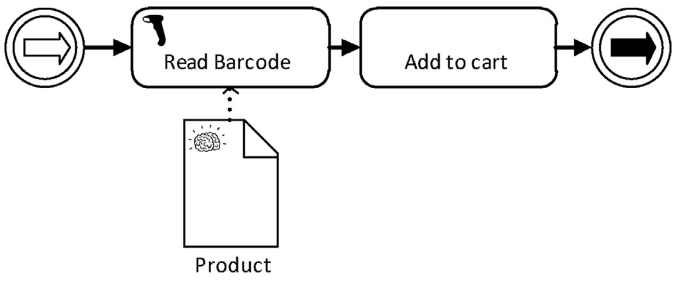

Fig. 6. Reader Task example.

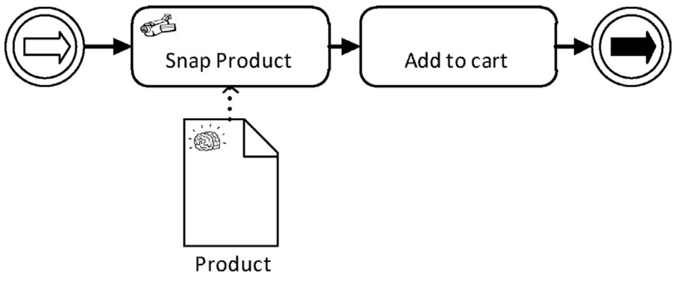

Fig. 7. Image Task example.

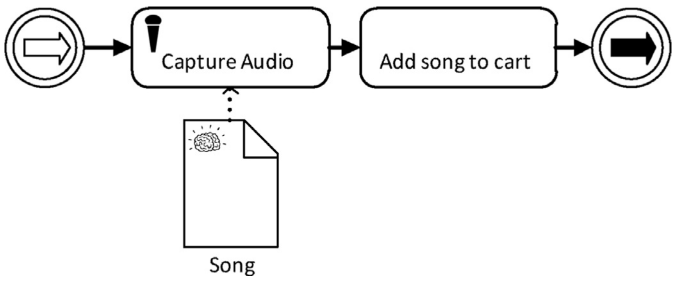

Fig. 8. Audio Task example.

- ***Audio Task***
An ***Audio Task*** is a *Task* that uses an audio stream to collect a particular information in the business environment. The data collected is an audio segment. An Audio Task object shares the same shape as the Task, which is a rectangle that has rounded corners. However, there is a microphone in the upper left corner of the shape indicating that the Task is of type Audio (see Fig. 4). In addition, the ***Audio Task*** has a unique connection to one Smart Object.
Example: The process participant hears music he likes. He wants to buy it but he neither knows the singer nor the title of the song. The user can add a song to his cart after providing a 10 s excerpt of it (e.g., Shazam). The ubiquitous business process fragment is shown in Fig. 8.
- ***Collector Task***
A ***Collector Task*** is a *Task* that collects a particular information in the business environment asides from using sensors, smart readers, video or audio feed. The collection is usually accomplished from databases, files or via a proxy entity (e.g., strangers take pictures in a party using distinct phones, they want to share the images, but without sharing their contact information or befriending each other on social networks, the collection can be accomplished from each others' phones by

[13] http://www.homeplus.co.kr.

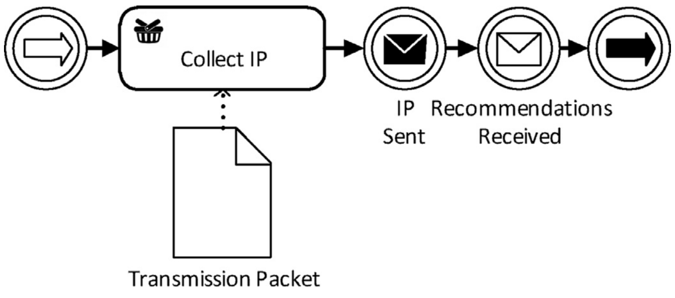

Fig. 9. Collector Task example.

having an ephemeral group context connection [23]). These are distinctive traits that supersede the capabilities of the Service Task. A Collector Task object shares the same shape as the Task, which is a rectangle with rounded corners. However, there is a hamper in the upper left corner of the shape to indicate that the Task is a Collector Task (see Fig. 4). In addition, the ***Collector Task*** can either be connected to a *Data Object* or a *Data Store* depending on the persistence of the data; ephemeral for the former and permanent for the latter.
Example: The process participant is on a trip and she wants to watch the highest rated movies in her current location (e.g., Movie Suggestions for Netflix, Video Recommendation for YouTube). Location, in addition to GPS sensors, can also be determined through the carrier network. The list of the highest rated movies in the user's current location is displayed to her after collecting her current location. The ubiquitous business process fragment is shown in Fig. 9.

4.2.3. Events

An Event is something that occurs during the course of a process and affects its flow. BPMN v2.0 defines twelve Events, i.e., None, Message, Timer, Error, Escalation, Cancel, Compensation, Conditional, Link, Signal, Terminate Multiple and Parallel Multiple (None has no trigger, therefore no `EventDefinition`, while Multiple and Parallel Multiple have multiple triggers as indicated on the Meta-Model). The twelve Events are classified into three types; Start, Intermediate (Catch and Throw) and End. In addition, they are two flavors of events: Events that catch a trigger and Events that throw a result. For that matter, Start and Intermediate Catch belong to the former category while Intermediate Throw and End belong to the latter one. Hereafter, we focus on Events that catch a trigger (from the business environment).

Building on the previous classifications, uBPMN adds five new events that catch a trigger to the BPMN v2.0 standard. These are Sensor, Reader, Image, Audio, and Collector. The uBPMN additional Events can be used at any process level. Similar to the events Message, Escalation, Error, Signal and Multiple, uBPMN supplementary events do indeed carry data. Note, they carry the same names as the uBPMN additional Tasks. This is because they follow the same concept as the BPMN Message (Tasks and Events) where in Tasks the process participant voluntarily takes action while in Events the action is triggered once the event is fired. Although being a convention rather than a rule, Events labels are framed in passive voice (e.g., Code Scanned) while Activity labels are expressed in active voice (e.g., Scan Code).

The Sensor, Reader, Image, Audio, and Collector Events reflect the uBPMN extension of the Flow Object Event following the structure of the meta-model presented in Figs. 1 and 2, the XSD of Listing 1 and the guidelines explained below:

- ***Sensor Events***
  A ***Sensor Event*** is an *Event* triggered by a particular information sensed in the business environment. There are two variations: Start Event and Catch Intermediate Event as shown in Fig. 10. Unlike the ***Sensor Task***, linking the ***Sensor Event*** to a ***Smart Object*** is possible (carries data and is of type start and/or intermediate catch; [42, p. 235]) but is not necessary since this is an event and it will be fired once the ***Smart Object*** presents itself in the business environment and the information is indeed sensed.
  Example: The process participant is in a smart home. The carbon dioxide sensor detects the level of CO_2 in the air and adjusts the air conditioning. The ubiquitous business process fragment is shown in Fig. 11. Note, we use a Sensor Start Event because the collaboration process (a process with two or more participants) with the smart home starts when the participant walks in.
- ***Reader Events***
  A ***Reader Event*** is an *Event* triggered by a particular type of information read in the business environment. There are two variations: Start Event and Catch Intermediate Event as shown in Fig. 10. Unlike the ***Reader Task***, linking the ***Reader Event*** to a ***Smart Object*** is possible (carries data and is of type start and/or intermediate catch; [42, p. 235]) but is not mandatory since this is an event and it will be fired once the ***Smart Object*** presents itself in the business environment and the information is indeed read.
  Example: The process participant is about to exit the highway. When vehicles pass through any of the toll plazas on the highway exits, an RFID reader scans the RFID tags installed on the vehicles. The data is then sent to a server to deduct the toll from the vehicle/tag owner's balance (i.e., e-Tolling). The ubiquitous business process fragment is shown in Fig. 12. Note, we

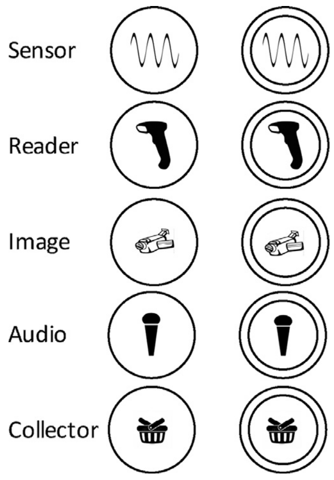

Fig. 10. uBPMN events.

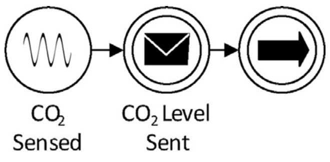

Fig. 11. Sensor Event example.

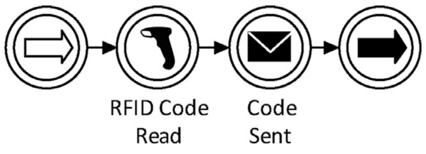

Fig. 12. Reader Event example.

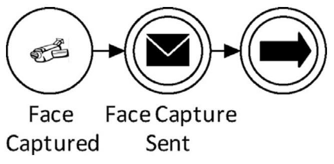

Fig. 13. Image Event example.

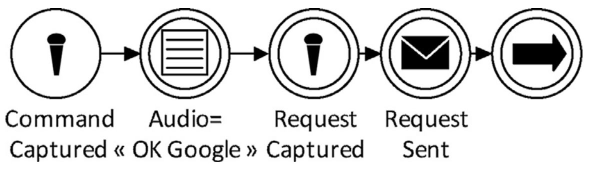

Fig. 14. Audio Event example

use a Reader Intermediate Event of type Catch because the process started when the driver entered the highway.

- ***Image Events***
  An ***Image Event*** is an *Event* triggered by a particular information snapped in the business environment. There are two variations: Start Event and Catch Intermediate Event as shown on Fig. 10. Unlike the ***Image Task***, linking the ***Image Event*** to a ***Smart Object*** is possible (carries data and is of type start and/or intermediate catch; [42, p.235]) but is not mandatory since this is an event and it will be fired once the ***Smart Object*** presents itself in the business environment and the information is indeed snapped.
  Example: A regular customer walks into a restaurant. The system, after taking a picture of the customer's face at the entrance, sends it to the clerk to prepare his usual order right away (e.g., Facedeals by Facebook[14]). The ubiquitous business process fragment is shown in Fig. 13. Note, we use an Image Start Event because the order process starts when the regular customer's face is captured and identified.
- ***Audio Events***
  An ***Audio Event*** is an *Event* triggered by an audio information captured in the business environment. There are two variations: Start Event and Catch Intermediate Event as shown on Fig. 10. Unlike the Audio Task, linking the Audio Event to a Smart Object via a Data Association is possible (carries data and is of type start and/or intermediate catch; [42, p. 235]) but is not mandatory since this is an event and it will be fired once the Smart Object presents itself in the business environment and the information is indeed captured.
  Example: The process participant wants to buy a product via audio commands (e.g., "*OK Google*" then "*request*" for Google Now, "*Alexa*" then "*request*" for Amazon Echo, "*Siri*" then "*request*" for Apple Siri; *request* changes as appropriate). For example, "*OK Google*" then "*Flights to San Francisco*" will list flights to San Francisco (the outbound location is sensed or collected by the system). The ubiquitous business process fragment is shown in Fig. 14. Note, the process starts with the audio capture which explains the use of the Audio Start Event (here it is "OK Google"). The Audio Intermediate Event of type Catch will capture the request (here it is "Flights to San Francisco").
- ***Collector Events***
  A ***Collector Event*** is an *Event* triggered by a particular information collected from a database or a file in the business environment (e.g., Events on a DBMS, a file recently synchronized on a remote storage such as Dropbox). There are two variations: Start Event and Catch Intermediate Event as shown on Fig. 10. Unlike the Collector Task, linking the Collector Event to a Smart Object via a Data Association is possible (carries data and is of type start and/or intermediate catch; [42, p. 235]) but is not mandatory since this is an event and it will be fired once the Smart Object presents itself in the business environment and the information is indeed collected.
  Example: Dat Autohus is a large German used-car dealer. Each day many cars are accepted and prepared for resale. All vehicles are parked on a large lot. If a prospective customer goes on a test drive, it is not necessary that the car gets returned to its original spot when the test drive is done. To keep track of its vehicles, the company logs the parking locations and sends a notification with the new parking spot to each potential buyer [2]. The ubiquitous business process fragment is shown in Fig. 15. Note, the notification process starts when location changes on Dat Autohus logs which explains the use of the Collector Start Event.

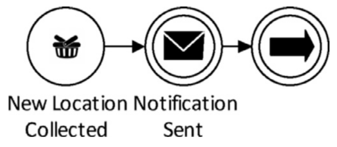

Fig. 15. Collector Event example

5. Illustrative example

Inspired from Trylive[15] and HappyView[16], this section presents an overall example in which we show how uBPMN is used to describe the ubiquitous business process whose business objective is "*Order Eyeglass Frames*". We limit ourselves to the most ubiquitous segment of the process and assume that the customer owns an account with details such as gender and age already on record (we neither cover the account creation process nor the checkout process in this example). The scenario of "*Order Eyeglass Frames*" process is presented as follows:

1. Once the potential customer signs in, her current location is determined (Sensor Task and/or Collector Task depending on the device) and reported to the server.
2. Based on the location information reported, gender and age, a list of the most popular items are recommended to the user.
3. The user tries as many eyeglass frames as she wants before adding them to the cart.

Building on the scenario, we can stress two points. First, because both desktop and mobile devices are targeted, the process must be *location-independent*. Second, the interactions with the business environment are accomplished *with the least number of human interventions* (i.e., capture current location, age, gender and face of the user). The two key pillars confirm that "*Order Eyeglass Frames*" is in fact a ubiquitous business process. The first business rule mandates that the current location of the user should be collected and reported to the backend of the system. In the same vein, the third business rule offers the possibility to virtually try on the frames before adding them to the shopping cart. From a ubiquitous computing standpoint, these two rules respectively invoke the technologies of context and augmented reality. BPMN v2.0 neither offers a mechanism to collect and quantify the location data nor can it represent an input video stream. As a result, we have to use uBPMN to describe these ubiquitous business rules.

[14] http://www.dailymail.co.uk/sciencetech/article-2187801/Were-watching-The-camera-recognise-Facebook-picture-time-walk-shop.html.

[15] http://www.trylive.com/.

[16] http://www.happyview.fr/.

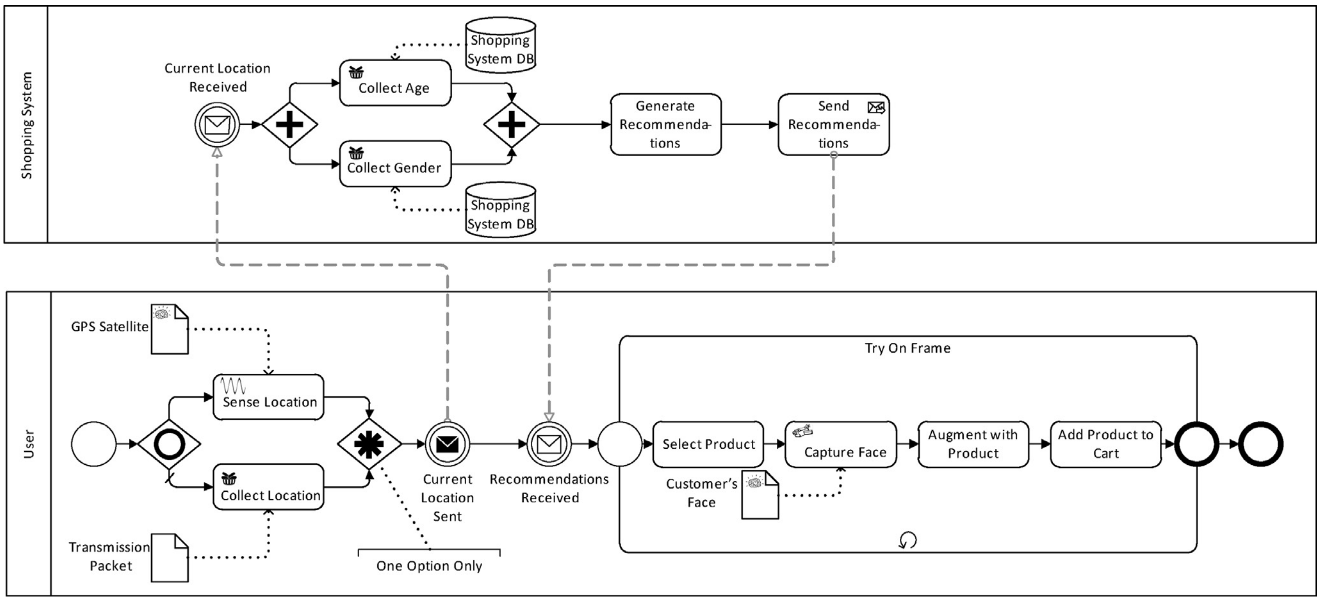

Fig. 16. *Order Eyeglass Frames* ubiquitous business process diagram in uBPMN (DB: Data-Base).

The diagram of "*Order Eyeglass Frames*" is shown in Fig. 16. The Complex Gateway is placed to limit capturing location to one medium; either sensing from the GPS satellites or collecting from the transmission packets (whichever option is efficient). That segment is a simplified version of Google Fused Location[17] that extends the battery life and guarantees the sustainability feature of ubicomp. It also fulfills the requirement of running the process via both mobile (either sensing or collecting) and desktop (only collecting) devices. Note, even though mobile devices, sensing location can be substituted by collecting location. Still, the location sensing frequency can be adapted to the activity sensing result (e.g., Google Activity Recognition[18]). In addition to sustainability and the inherent AIDC, context awareness and augmented reality can also be depicted in the process. The former is expressed through recommending the most popular items based on the context of the potential customer (gender, age, location) while the latter is about enabling her/him to try on the frames virtually via the camera of the device.

6. Discussion

This paper is both a specification document and a modeling guide for uBPMN. We introduce uBPMN to address the many cases in the field of ubiquitous computing that supersede the capabilities of BPMN v2.0. The limitation of BPMN v2.0 (*i*) hinders the dissemination of ubicomp ideas within business process management and (*ii*) blocks any verification, validation and transformation initiative that may arise within ubiquitous business processes.

The extension from BPMN v2.0 to uBPMN was accomplished following the process adopted by many references that targeted extending BPMN. We also adopt the same phrasing pattern used by the Object Management Group in order to help BPMN users easily understand the idea behind uBPMN. In doing so, we lay the groundwork to tackle the model of a ubiquitous business process rather than focusing only on its visual diagram (a diagram is only a perspective of a model). So first, we highlight the requirements of ubiquitous computing with respect to business process modeling and how the inability of BPMN to represent ubicomp input technologies results in not being able to fully support ubiquitous business processes. Second, and building on the ubicomp requirements highlighted, we extend the core structure of BPMN. This comes up in two representations; a MOF meta-model and an XSD. We then expand the notation with new core modeling elements following the core structure of uBPMN. Each and every core modeling element is presented along a concrete example from the business world.

Additionally, an illustrative example of a ubiquitous business process whose business objective is "*Order Eyeglass Frames*" is provided. The illustrative example process deploys three ubicomp input technologies (image, collector and sensor) and depicts three ubicomp features (sustainability, augmented reality and context awareness) in addition to the inherent AIDC. To sum up, uBPMN satisfies the requirements of ubiquitous computing, and serves as a stepping stone towards an official extension of BPMN v2.0 with the purpose of modeling ubiquitous business processes.

The ten examples presented across Section 4 along with the illustrative example in Section 5 *are all real and appear in the business world*. They highlight how uBPMN can appropriately model those concrete ubiquitous business rules. These were managed in an ad hoc fashion with random digressions from the standard. uBPMN can also maximize the visual quality of ubiquitous business process models. When modeling, process analysts can reduce ambiguity by placing the most meaningful core modeling element(s) that best describe(s) the ubiquitous business rule(s) instead of burdening the diagrams with superfluous text annotations. When implementing the models, process engineers can accurately convey the ideas of the analysts which reduces the round-trips between analysts and engineers, and, by extension, the time and cost of doing so.

This paper can be a reference for entities who want to model, verify, validate and/or transform ubiquitous business processes. The MOF meta-model along with the XSD can facilitate that purpose and be loaded, for example, into the Eclipse development environment with the appropriate plugins. Both the meta-model and the XSD were extended according to the Object Management Group guidelines, further indicating their validity. Finally, we also validated the XSD using a well-known tool.

[17] https://developers.google.com/android/reference/com/google/android/gms/location/FusedLocationProviderApi.

[18] http://developer.android.com/reference/com/google/android/gms/location/ActivityRecognition.html.

An important limitation of our approach is that ubicomp is progressing at an exponential pace. This version of uBPMN represents the current state of ubicomp input technologies. As ubicomp evolves and new input technologies/features are developed, there will need to be future extensions that will offer an even better user experience. To support this, it is important to bear in mind that uBPMN is extensible itself.

7. Conclusion and future work

In this paper, we propose a specification of uBPMN, which is an extension of BPMN v2.0 that allows for modeling ubiquitous business processes. In light of this, we conservatively extend BPMN to address the many shortcomings that it exhibits when it comes to describing ubiquitous business processes. uBPMN progresses BPMN v2.0 by:

- Accurately modeling ubiquitous business processes via a set of more meaningful core modeling elements that better describe ubiquitous business rules.
- Helping process personnel to represent, understand, and convey the ubiquitous business processes in a straightforward fashion.
- Laying the groundwork to deploying a myriad of ubiquitous computing capabilities such as context awareness, augmented reality, sustainability and ambient intelligence.
- Being the reference of any Validation and Verification and transformation initiative of uBPMN models.

To evolve the contribution of this paper, we plan on keeping up with the advances of ubicomp and conducting separate experiments to study its impact on business process management as well as the completeness of uBPMN. We also plan on proposing a verification and validation technique for uBPMN models. Afterwards, we aim to tackle the transformation of uBPMN models into working code and extend a framework that was partly built by our research team at the Human-Computer Interaction Institute, Carnegie Mellon University. The framework is named AWARE and is accessible via *awareframework.com*.

References

[1] G. Abowd, A. Dey, P. Brown, N. Davies, M. Smith, P. Steggles, Towards a better understanding of context and context-awareness, in: Handheld and Ubiquitous Computing, Springer, 1999, pp. 304–307.
[2] S.A. Ahson, M. Ilyas, RFID Handbook: Applications, Technology, Security, and Privacy, CRC press, 2008.
[3] N. Aoumeur, J. Fiadeiro, C. Oliveira, Towards an architectural approach to location-aware business process, in: Proceedings of the Thirteenth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2004. WET ICE 2004, IEEE, 2004, pp. 147–152.
[4] S. Appel, P. Kleber, S. Frischbier, T. Freudenreich, A. Buchmann, Modeling and execution of event stream processing in business processes, Inf. Syst. 46 (2014) 140–156.
[5] R.T. Azuma, et al., A survey of augmented reality, Presence 6 (4) (1997) 355–385.
[6] C. Bauer, C. Strauss, C. Stummer, A. Trieb, Context-aware services in cooperative value chains: a key player-centred approach, Scal. Comput. Pract. Exp. 14 (3) (2013) 139–154.
[7] J. Becker, P. Delfmann, A. Dreiling, R. Knackstedt, D. Kuropka, Configurative process modeling–outlining an approach to increased business process model usability, in: Proceedings of the Fifteenth IRMA International Conference, 2004, pp. 1–12.
[8] J. Becker, M. Rosemann, C. von Uthmann, Guidelines of business process modeling, in: Business Process Management, Springer, 2000, pp. 30–49.
[9] P. Bocciarelli, A. D'Ambrogio, A bpmn extension for modeling non functional properties of business processes, in: Proceedings of the 2011 Symposium on Theory of Modeling and Simulation: DEVS Integrative M&S Symposium, Society for Computer Simulation International, 2011, pp. 160–168.
[10] R. Braun, Behind the scenes of the bpmn extension mechanism principles, problems and options for improvement, in: Proceedings of the 2015 Third International Conference on Model-Driven Engineering and Software Development (MODELSWARD),, IEEE, 2015, pp. 1–8.
[11] R. Braun, W. Esswein, Classification of domain-specific bpmn extensions, in: The Practice of Enterprise Modeling, Springer, 2014, pp. 42–57.
[12] R. Braun, H. Schlieter, Requirements-based development of bpmn extensions: The case of clinical pathways, in: Proceedings of the 2014 IEEE First International Workshop on the Interrelations between Requirements Engineering and Business Process Management (REBPM),, IEEE, 2014, pp. 39–44.
[13] A. Bucchiarone, A. Marconi, M. Pistore, H. Raik, Captevo: Context-aware adaptation and evolution of business processes, in: Proceedings of the Service-Oriented Computing-ICSOC 2011 Workshops, Springer, 2012, pp. 252–254.
[14] A. Bucchiarone, A. Marconi, M. Pistore, H. Raik, Dynamic adaptation of fragment-based and context-aware business processes, in: Proceedings of the 2012 IEEE Nineteenth International Conference on Web Services (ICWS), IEEE, 2012, pp. 33–41.
[15] J.F. Chang, Business Process Management Systems: Strategy and Implementation, CRC Press, 2005.
[16] Y. Charalabidis, Revolutionizing Enterprise Interoperability through Scientific Foundations, IGI Global, 2014.
[17] S. Consolvo, D.W. McDonald, T. Toscos, M.Y. Chen, J. Froehlich, B. Harrison, P. Klasnja, A. LaMarca, L. LeGrand, R. Libby, et al., Activity sensing in the wild: a field trial of ubifit garden, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ACM, 2008, pp. 1797–1806.
[18] M. Cortes-Cornax, S. Dupuy-Chessa, D. Rieu, N. Mandran, Evaluating the appropriateness of the bpmn 2.0 standard for modeling service choreographies: using an extended quality framework, Softw. Syst. Model. 15 (2014) 1–37.
[19] K. Czarnecki, S. Helsen, Classification of model transformation approaches, in: Proceedings of the Second OOPSLA Workshop on Generative Techniques in the Context of the Model Driven Architecture, USA, vol. 45, 2003, pp. 1–17.
[20] K. Czarnecki, S. Helsen, Feature-based survey of model transformation approaches, IBM Syst. J. 45 (3) (2006) 621–645.
[21] T. da Cunha Mattos, F.M. Santoro, K. Revoredo, V.T. Nunes, A formal representation for context-aware business processes, Comput. Ind. 65 (8) (2014) 1193–1214.
[22] M. Das, M. Deb, M. Wilkins, Oracle Business Process Management Suite 11g Handbook, McGraw-Hill Osborne Media, 2011.
[23] A.A. de Freitas, A.K. Dey, The group context framework: An extensible toolkit for opportunistic grouping and collaboration, in: Proceedings of the Eighteenth ACM Conference on Computer Supported Cooperative Work & Social Computing, ACM, 2015, pp. 1602–1611.
[24] M. Dumas, M. La Rosa, J. Mendling, H.A. Reijers, Fundamentals of Business Process Management, Springer, 2013.
[25] R. Ebad, Ubiquitous computing: A brief review of impacts and issues, Scal. Comput. Pract. Exp. 2 (3) (2014) 59–68.
[26] E. Fleisch, Business perspectives on ubiquitous computing, M-Lab Work. Pap. (4) (2001) 83–87.
[27] J. Friedenstab, C. Janiesch, M. Matzner, O. Muller, Extending bpmn for business activity monitoring, in: Proceedings of the 2012 Forty Fifth Hawaii International Conference on System Science (HICSS), IEEE, 2012, pp. 4158–4167.
[28] M. Friedewald, O. Raabe, Ubiquitous computing: An overview of technology impacts, Telemat. Inf. 28 (2) (2011) 55–65.
[29] J. Froehlich, T. Dillahunt, P. Klasnja, J. Mankoff, S. Consolvo, B. Harrison, J.A. Landay, Ubigreen: investigating a mobile tool for tracking and supporting green transportation habits, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ACM, 2009, pp. 1043–1052.
[30] D. Gagne, A. Trudel, Time-bpmn, in: Proceedings of the IEEE Conference on Commerce and Enterprise Computing CEC'09, IEEE, 2009, pp. 361–367.
[31] P. Giner, V. Torres, V. Pelechano, Jisbd2007-06: Building ubiquitous business process following an mdd approach, Lat. Am. Trans. IEEE Rev. IEEE Am. Lat. 6 (4) (2008) 347–354.
[32] J.-Y. Jung, J. Kong, J. Park, Service integration toward ubiquitous business process management, in: Proceedings of the IEEE International Conference on Industrial Engineering and Engineering Management, IEEM 2008, IEEE, 2008, pp. 1500–1504.
[33] H. Kopetz, Internet of things, in: Real-Time Systems, Springer, 2011, pp. 307–323.
[34] M. Kuniavsky, Smart Things: Ubiquitous Computing User Experience Design: Ubiquitous Computing User Experience Design, Elsevier, 2010.
[35] M. Lijding, N. Meratnia, H. Benz, A. Matysiak Szóstek, Smart signs show you the way, I/O Vivat 22 (4) (2007) 35–38.
[36] T. Liu, P. Bahl, I. Chlamtac, Mobility modeling, location tracking, and trajectory prediction in wireless atm networks, IEEE J. Sel. Areas Commun. 16 (6) (1998) 922–936.
[37] A. Lodhi, V. Küppen, G. Saake, An extension of bpmn meta-model for evaluation of business processes, Sci. J. Riga Tech. Univ. Comput. Sci. 43 (1) (2011) 27–34.
[38] D.G. Lowe, Distinctive image features from scale-invariant keypoints, Int. J. Comput. Vis. 60 (2) (2004) 91–110.
[39] R. Martinho, D. Domingos, Quality of information and access cost of iot resources in bpmn processes, Proc. Technol. 16 (2014) 737–744.
[40] S. Meyer, A. Ruppen, C. Magerkurth, Internet of things-aware process modeling: integrating iot devices as business process resources, in: Advanced Information Systems Engineering, Springer, 2013, pp. 84–98.
[41] H. Mili, G. Tremblay, G.B. Jaoude, É. Lefebvre, L. Elabed, G.E. Boussaidi, Business process modeling languages: Sorting through the alphabet soup, ACM Comput. Surv. (CSUR) 43 (1) (2010) 4.
[42] OMG, Business Process Model and Notation 2.0, Technical Report, Object Management Group, Washington DC, USA, 2011.

[43] E. Pascalau, G.J. Nalepa, K. Kluza, Towards a better understanding of context-aware applications, in: Proceedings of the 2013 Federated Conference on Computer Science and Information Systems (FedCSIS), IEEE, 2013, pp. 959–962.
[44] R.M. Pillat, T.C. Oliveira, P.S. Alencar, D.D. Cowan, Bpmnt: A bpmn extension for specifying software process tailoring, Inf. Softw. Technol. 57 (2015) 95–115.
[45] K. Ploesser, J.C. Recker, M. Rosemann, Building A Methodology For Context-Aware Business Processes: Insights from an Exploratory Case Study, Association for Information Systems, 2010.
[46] D. Preuveneers, J. Van den Bergh, D. Wagelaar, A. Georges, P. Rigole, T. Clerckx, Y. Berbers, K. Coninx, V. Jonckers, K. De Bosschere, Towards an extensible context ontology for ambient intelligence, in: Ambient intelligence, Springer, 2004, pp. 148–159.
[47] T. Rademakers, Activiti in Action: Executable Business Processes in BPMN 2.0, Manning Publications Co., 2012.
[48] J. Recker, M. Rosemann, M. Indulska, P. Green, Business process modeling-a comparative analysis, J. Assoc. Inf. Syst. 10 (4) (2009) 1.
[49] M. Reichert, B. Weber, Business process compliance, in: Enabling Flexibility in Process-Aware Information Systems, Springer, 2012, pp. 297–320.
[50] W. Richard, A sensor classification scheme, IEEE Trans. Ultrason. Ferroelectr. Freq. Control 34 (2) (1987) 124–126.
[51] A. Rodríguez, A. Caro, C. Cappiello, I. Caballero, A BPMN Extension for Including Data Quality Requirements in Business Process Modeling, Springer, 2012.
[52] A. Rodríguez, E. Fernández-Medina, M. Piattini, A bpmn extension for the modeling of security requirements in business processes, IEICE Trans. Inf. Syst. 90 (4) (2007) 745–752.
[53] M. Roman, J. Al-Muhtadi, B. Ziebart, R. Campbell, M.D. Mickunas, System support for rapid ubiquitous computing application development and evaluation, in: Proceedings of the System Support for Ubiquitous Computing Workshop (UbiSys' 03) in Conjunction with UbiComp, vol. 3, 2003.
[54] A. Rozinat, W.M. van der Aalst, Conformance testing: Measuring the fit and appropriateness of event logs and process models, in: Proceedings of the Business Process Management Workshops, Springer, 2006, pp. 163–176.
[55] S. Sadiq, G. Governatori, K. Namiri, Modeling control objectives for business process compliance, in: Business Process Management, Springer, 2007, pp. 149–164.
[56] K. Saeedi, L. Zhao, P.R.F. Sampaio, Extending bpmn for supporting customer-facing service quality requirements, in: Proceedings of the 2010 IEEE International Conference on Web Services (ICWS), IEEE, 2010, pp. 616–623.
[57] M. Salatino, E. Aliverti, jBPM5 Developer Guide, Packt Publishing, 2012.
[58] M. Schultz, M. Radloff, Modeling concepts for internal controls in business processes–an empirically grounded extension of bpmn, in: Business Process Management, Springer, 2014, pp. 184–199.
[59] A. Sears, J.A. Jacko, Human-Computer Interaction Fundamentals, CRC Press, 2009.
[60] A.D. Smith, F. Offodile, Information management of automatic data capture: an overview of technical developments, Inf. Manag. Comput. Secur. 10 (3) (2002) 109–118.
[61] L.J.R. Stroppi, O. Chiotti, P.D. Villarreal, A bpmn 2.0 extension to define the resource perspective of business process models, in: Proceedings of the XIV Iberoamerican Conference on Software Engineering, 2011.
[62] L.J.R. Stroppi, O. Chiotti, P.D. Villarreal, Extending bpmn 2.0: method and tool support, in: Business Process Model and Notation, Springer, 2011, pp. 59–73.
[63] C.T. Sungur, P. Spiess, N. Oertel, O. Kopp, Extending bpmn for wireless sensor networks, in: Proceedings of the 2013 IEEE Fifteenth Conference on Business Informatics (CBI), IEEE, 2013, pp. 109–116.
[64] B.H. Thacker, S.W. Doebling, F.M. Hemez, M.C. Anderson, J.E. Pepin, E.A. Rodriguez, Concepts of Model Verification and Validation, Technical Report, Los Alamos National Lab., Los Alamos, NM (US), 2004.
[65] S. Tranquillini, P. Spieß, F. Daniel, S. Karnouskos, F. Casati, N. Oertel, L. Mottola, F.J. Oppermann, G.P. Picco, K. Römer, et al., Process-based design and integration of wireless sensor network applications, in: Business Process Management, Springer, 2012, pp. 134–149.
[66] M. Weiser, The computer for the 21st century, Sci. Am. 265 (3) (1991) 94–104.
[67] A.D. Wilson, Sensor-and recognition-based input for interaction, Hum. Comput. Interact. (2007) 153.
[68] C. Wolter, M. Menzel, A. Schaad, P. Miseldine, C. Meinel, Model-driven business process security requirement specification, J. Syst. Archit. 55 (4) (2009) 211–223.
[69] A. Yousfi, A. de Freitas, A. Dey, R. Saidi, The use of ubiquitous computing for business process improvement, IEEE Trans. Serv. Comput. (2015a), doi:10.1109/TSC.2015.2406694. in press.
[70] A. Yousfi, A.K. Dey, R. Saidi, J.-H. Hong, Introducing decision-aware business processes, Comput. Ind. 70 (2015b) 13–22.
[71] A. Yousfi, R. Saidi, A.K. Dey, Variability patterns for business processes in bpmn, Inf. Syst. e-Bus. Manag. (2015c) 1–25, doi:10.1007/s10257-015-0290-7.
[72] X. Zhu, J. Recker, G. Zhu, F. Maria Santoro, Exploring location-dependency in process modeling, Bus. Process Manag. J. 20 (6) (2014) 794–815.