2009 Fourth International Conference on Computer Sciences and Convergence Information Technology

BPMN and its Semantics for Information Management in Emergency Care

Patrik Eklund
Umeå University
Department of Computing Science
SE-90187 Umeå Sweden
peklund@cs.umu.se

Mats Johansson
Umeå University
Department of Computing Science
SE-90187 Umeå Sweden
matsj@cs.umu.se

Johan Karlsson
University of Málaga
Department of Computer Architecture
E-29080 Málaga, Spain
tjkarlsson@uma.es

Rickard Åström
Umeå University
Department of Computing Science
SE-90187 Umeå Sweden
rickarda@cs.umu.se

Abstract

Emergency and crisis response such as appearing in large traffic accident management involves coordination of emergency and rescue services and police. The management of information is a complex task where decision support on site and distance management requires a well-founded understanding of the underlying processes. The Business Process Modeling Notation (BPMN) is used for encoding processes related to emergency medical services. BPMN is basically a syntax where the semantic part is left to be specified by the applications. Our approach to semantics is drawn from categorical approaches to representing signatures and generalized terms. In particular, monads over `Set`*, the category of sets, composed with the term monad (over* `Set`*) turn out to be useful for modeling uncertainties both with respect to representation of observations as well as time. BPMN Flow Objects, Connecting Objects as well as Artifacts are represented as suitable morphisms in the underlying categories enabled by the monadic instrumentation.*

1 Introduction

In this paper we will present a category theory based semantics for BPMN used in cross-functional and event-based workflow modeling of emergency medical services, fire/rescue services and police, acting within and by a coordinated management[1]. Our example is drawn from an accident and disaster scenario for a traffic accident on a highway affecting many cars and and involving severe injuries. In addition, the accident may have been initially caused by a lorry transporting dangerous chemicals. This provides a challenge for the core emergency and rescue services where also police with traffic management must be involved (Figure 1).

Our modeling approach will focus in particular on emergency medical services involving ambulances and emergency care.

There are several guidelines for emergency care, and these guidelines are often developed regionally. National guidelines exist but are shallow and provide only overviews typically with no specific information for on-site emergency care. Well developed regional guidelines (e.g. [1]) are fully adequate for emergency carers but are mostly insufficient for information and decision support system development.

The purpose of this paper is to provide some initial steps towards improving this situation by describing an information representation formalism which enables management of uncertainty as well as information composition in cross-functional scenarios.

2 Trauma

The emergency care guidelines cover e.g. respiratory syndromes, circulatory failure, trauma and prenatal situations. We will focus on trauma which typically may involve skull fracture and spinal cord injury. It could also involve facial, thoracic and/or abdominal injuries, or bleeding, hypothermia and burn wounds.

[1]Research presented in this paper is supported by the Nordic Safety and Security (NSS, 2008-2011) project funded by the European Regional Development Fund (ERDF)

978-0-7695-3896-9/09 $26.00 © 2009 IEEE
DOI 10.1109/ICCIT.2009.27

The guidelines [1] for trauma/accident follow the principles of Pre-Hospital Trauma Life Support (PHTLS) and starts by describing generally the process on the site of the accident. Responsibilities are defined and steps to take involve making an assessment of the situation and setting up goals for the care. Resources are acquired and the rescue actions are coordinated with other emergency and police actions.

The emergency medical care *core steps* involves

- first assessments
- immediate life saving actions
- prioritization of injuries
- securing important functions before transportation
- minimizing complications during transport
- distribution of injured over health care units

Evaluations of injuries also come with classifications like very urgent, urgent and not urgent. There are general criteria for severe trauma, and care of specific traumas are guided by *detailed descriptions* and recommendations concerning

- cause of the injury
- anamnesis
- symptoms
- investigation
- intervention
- monitoring
- transportation

Note e.g. how investigation overlaps with intervention, and monitoring overlaps with transportation.

The core steps and detailed description are continuously documented (BPMN artifact `Doc` in Figure 2), and the content of `Doc` is updated and refined both with respect to data as well as certainty values attached to data. From a formal point of view, data is represented by signatures and terms modeled by the term monad, and certainties by many-valuedness as modeled by the many-valued powerset monad. The composition of the many-valued powerset monad with the term monad provides the underlying data structure for `Doc`. The BPMN Sub-Process for the emergency care core steps is outlined in Figure 2.

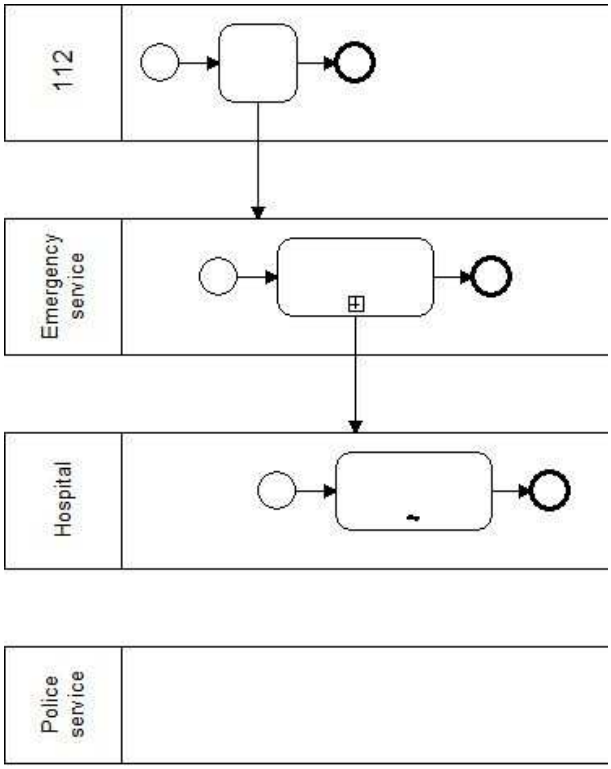

Figure 1. General view of BPMN Sub-Processes and Sequence Flow between coordinated units represented by BPMN Pools.

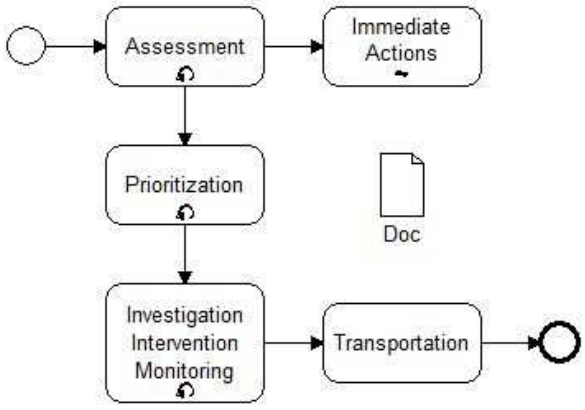

Figure 2. The emergency medical care Sub-Process reflecting the guideline.

3 Business Process Modeling Notation (BPMN)

BPMN [3] as a process language is intended to be a a standardized bridge for the gap between process design and process implementation. Process design must be closer to human language, whereas process implementation in closer to computer language. BPMN comes with syntax but possess no semantics. BPMN enables not just process information but in particular the communication of process information between participants in the process. Indeed, nothing is said about semantics, and therefore clearly nothing is specified concerning logic and the way participants act and infer with and by tasks and sub-process in the process as a whole.

Our standpoint is that different logics will occur with one and the same process. Here we take logic in the general meaning according to the formalism of general logics [20]. Within one atomic task there is just one underlying logic since an atomic task typically contains specific guidelines and rules based on which patterns of actions and methods of inference emerge and develop. A task typically then contains a rule base and a sub-process a structure of 'communication logics'. Whenever inferences are made, typically using the inference rules attached to the entailment attached with the task, the result is always and at least a variable substitution. It is more specific to consider the whole variable substitution as an artifact than just the data attached to variables. This seems like an unnecessary subtle distinction, but it is important to note that sub substitutions can be composed whereas there are no corresponding operations for enlarging just data.

BPMN diagrams build syntactically upon four basic categories of elements, namely Flow Objects, Connecting Objects, Artifacts and Swimlanes. Flow Objects, represented by Events, Activities and Gateways, define the behaviour of processes. Start and End are typical Event elements. Task and Sub-Process are the most common Activities. There are three Connecting Objects, namely Sequence Flow, Message Flow and Association. Swimlanes consist of Pools and Lanes. Gateways, as Event elements, handle branching, forking, merging, and joining of paths. Pragmatically it is easy to see most of the logic residing in gateways. However, we must distinguish between logic that is internal to the process and logic that resides more on the meta-level and semantically outside the tasks. Logic internal to Tasks and Sub-Processes are managed by general logics formalisms, whereas meta-level logic, which may even be informal, is executed in Gateways. A Data Object (like `Doc` is an Artifact and indeed not a Flow Object.

BPMN indeed provides no semantics but gives some hints where semantic issues eventually will become visible. A Sequence Flow is expected to *show the order that activities will be performed in a Process*, but nothing is said about overlaps, like the one for monitoring and transportation of patients. A Message Flow is expected to *show the flow of messages between two participants that are prepared to send and receive them*. Synchrony or asynchrony is not mentioned at this point.

4 Monads

A monad over a category $\mathtt{C}$ is a written $\mathbf{F} = (\mathsf{F}, \eta, \mu)$, where $\mathsf{F} : \mathtt{C} \longrightarrow \mathtt{C}$ is a covariant functor, and $\eta : \mathsf{id} \longrightarrow \mathsf{F}$ (id is the identity functor) and $\mu : \mathsf{F} \circ \mathsf{F} \longrightarrow \mathsf{F}$ are natural transformations such that

$$\mu \circ \mathsf{F}\mu = \mu \circ \mu\mathsf{F}$$

and

$$\mu \circ \mathsf{F}\eta = \mu \circ \eta\mathsf{F} = \mathsf{id}_{\mathsf{F}}$$

.

The Kleisli category $\mathtt{C}_{\mathbf{F}}$ for $\mathbf{F}$ (over $\mathtt{C}$) consists of the same objects as in $\mathtt{C}$ and morphisms $f\colon X \longrightarrow Y$ in $\mathtt{C}_{\mathbf{F}}$ are morphisms $f\colon X \longrightarrow \mathsf{F}Y$ in $\mathtt{C}$. Thus, $\eta^F_X\colon X \longrightarrow \mathsf{F}X$ in $\mathtt{C}$ represents the identity morphism in $\mathtt{C}_{\mathbf{F}}$. Composition of morphisms in $\mathtt{C}_{\mathbf{F}}$ is given by

$$(X \xrightarrow{f} Y) \diamond (Y \xrightarrow{g} Z) = X \xrightarrow{\mu^F_Z \circ \mathsf{F}g \circ f} \mathsf{F}Z.$$

4.1 The term monad

Let $\Omega = \bigcup_{n=0}^{\infty} \Omega_n$ be an operator domain, where Ω_n contains n-ary operators. The term functor $\mathsf{T}_\Omega\colon \mathtt{Set} \to \mathtt{Set}$ is given as $\mathsf{T}_\Omega X = \bigcup_{k=0}^{\infty} T^k_\Omega(X)$, where

$$\begin{aligned} T^0_\Omega(X) &= X, \\ T^{k+1}_\Omega(X) &= \{(n, \omega, (m_i)_{i\le n}) \mid \omega \in \Omega_n, m_i \in T^k_\Omega(X)\}. \end{aligned}$$

Note that $(n, \omega, (x_i)_{i\le n})$ represents the more common notation $\omega(x_1, \ldots, x_n)$ for a term.

In order to obtain the term monad ([19]) $\mathbf{T}_\Omega = (\mathsf{T}_\Omega, \eta^{T_\Omega}, \mu^{T_\Omega})$, define $\eta^{T_\Omega}_X(x) = x$, and let $\mu^{T_\Omega}_X = id^\star_{T_\Omega X}$ be the Ω-extension of $id_{T_\Omega X}$ with respect to $(T_\Omega X, (\sigma_{n\omega})_{(n,\omega)\in\Omega})$.

Morphisms in Kleisli categories are *generalized substitutions*. In the case of the monad being the term monad, generalized substitutions are clearly the ordinary substitutions of variables with terms.

4.2 The fuzzy powerset monad

The usual covariant powerset monad $\mathbf{P} = (\mathsf{P}, \eta, \mu)$ is given with $\mathsf{P}X$ being the set of subsets of X, together with $\eta_X(x) = \{x\}$ and $\mu_X(\mathcal{B}) = \bigcup \mathcal{B}$.

For the many-valued extension let L be a completely distributive lattice. In the case of $L = \{0, 1\}$ we write $L = 2$. The many-valued powerset functor L ([15]) is now defined by $\mathsf{L}X$ being the set of mappings $A : X \longrightarrow L$, where morphisms $f\colon X \longrightarrow Y$ in $\mathtt{Set}$ extend to morphisms $\mathsf{L}f$ according to

$$\mathsf{L}f(A)(y) = \bigvee_{f(x)=y} A(x).$$

We obtain a monad ([19]) $\mathbf{L} = (\mathsf{L}, \eta, \mu)$, when $\eta_X : X \longrightarrow \mathsf{L}X$ is given by

$$\eta_X(x)(x') = \begin{cases} 1 & \text{if } x = x' \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

and $\mu : \mathsf{L} \circ \mathsf{L} \longrightarrow \mathsf{L}$ is given by

$$\mu_X(\mathcal{M})(x) = \bigvee_{A \in \mathsf{L}X} A(x) \wedge \mathcal{M}(A). \tag{2}$$

Note now how $\mathbf{2}$ indeed is the usual covariant powerset monad $\mathbf{P}$. Further the category of sets and relations, where objects are sets and morphisms $f : X \longrightarrow Y$ are ordinary relations $f \subseteq X \times Y$ with composition of morphisms being relational composition, is isomorphic to the Kleisli category $\mathtt{Set}_{\mathbf{2}}$.

4.3 Variables substituted by many-valued sets of terms

The composition $\mathsf{L} \circ \mathsf{T}_\Omega$ can be extended to a monad $\mathbf{L} \bullet \mathbf{T}_\Omega$ by means of a 'swapping' ([8]) $\sigma_X : \mathsf{T}_\Omega \mathsf{L}X \to \mathsf{L}\mathsf{T}_\Omega X$ where $\sigma_{X\,|T^0\mathsf{L}X} = \mathsf{id}_{\mathsf{L}X}$, and further, for $l = (n, \omega, (l_i)_{i \le n}) \in T^\alpha \mathsf{L}X$, $\alpha > 0$, $l_i \in T^{\beta_i}\mathsf{L}X$, $\beta_i < \alpha$, let

$$\sigma_X(l)((n', \omega', (m_i)_{i \le n})) = \\ = \begin{cases} \bigwedge_{i \le n} \sigma_X(l_i)(m_i) & \text{if } n = n' \text{ and } \omega = \omega' \\ 0 & \text{otherwise} \end{cases}$$

The natural transformations $\eta\colon \mathsf{id} \longrightarrow \mathsf{LT}$ and $\mu\colon \mathsf{LTLT} \longrightarrow \mathsf{LT}$ are defined as follows

$$\eta = \eta^{\mathsf{L}\mathsf{T}} \circ \eta^{\mathsf{T}} \qquad \mu_X = \mathsf{L}\mu^{\mathsf{T}}_X \circ \mu^{\mathsf{L}}_{\mathsf{T}\mathsf{T}X} \circ \mathsf{L}\sigma_{\mathsf{T}X}$$

Morphisms $f\colon X \nrightarrow Y$ in $\mathtt{Set}_{\mathbf{L}\bullet\mathbf{T}_\Omega}$ capture the notion of variables being substituted by many-valued sets of terms. The Data Object `Doc` is such a morphism, i.e. $\mathtt{Doc} : X \longrightarrow \mathsf{L}\mathsf{T}_\Omega X$, where X contains `Skull fracture`, `Spinal Cord Injury`, and so on, and `Very Urgent`, `Urgent` and `Not Urgent` are included in the lattice L.

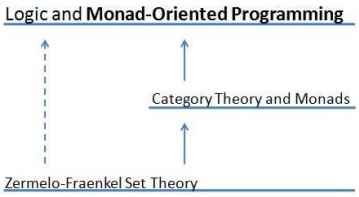

Figure 3. Monad-Oriented Programming (MOP).

4.4 Monad-Oriented Programming

In Section 3 we made a distinction between process integrated logic and meta-level logic.

For comparison, also in automata you could make this distinction. Let S be a set of states and E a set of events. A *state transition* is a mapping $t : S \times E \longrightarrow S$. The state transition function together with S and E is traditionally called a *deterministic automaton*, being *finite* if S is a finite set. The set E is seen as an alphabet and E^* as the set of 'words' created by the characters in the alphabet. Formally speaking, E^* is the free monoid over E. Kleene's theorem, about any language accepted by a finite automaton being regular, is folklore in this non-categorical setting where all sets are assumed to reside in the underlying set theory. This means that sets appear both in the object language as well as the metalanguage (of set theory) when discussing events and states.

Monad-oriented programming clearly means making the distinction. For automata this means e.g. using the powerset monad when managing non-deterministic automata, and not just the powerset of S as an element on the meta-level of set theory. Using monads e.g. in general logics means placing the language core elements within a categorical framework thereby at the same time identifying which part of the language remains on the meta-level, typically represented formally by Zermelo-Fraenkel set theory.

BPMN being syntactic means that simulation and animation of processes are not straightforward as underlying computational models, and their semantics, are (deliberately) missing. Discrete simulation models, such as e.g. provided by Petri nets, are expected to be useful. At the same time we must be aware of the generality as provided by the partially ordered monads and their compositions, as the underlying monad for conventional computations is just the term monad from declarative point of view and the powerset monad from relational point of view. The use of a wide range of partially ordered monads indeed goes far beyond just using the conventional monads.

5 A Kleene algebra of substitutions

A partially ordered monad $\mathbf{F} = (\mathsf{F}, \preceq, \eta, \mu)$ consists of a monad (F, η, μ), where $(\mathsf{F}X, \preceq)$ is a partially order set and certain conditions must be fulfilled. See [13] for detail. In [10] we showed that $\mathbf{L} \bullet \mathbf{T}_\Omega$ can be extended to a partially ordered monad. We further showed that $(Hom(X, \mathsf{FT}_\Omega X), +, \cdot, ^*, 0, 1)$, i.e. the set of morphisms in the Kleisli category $\mathtt{Set}_\mathbf{F}$, is a Kleene algebra ([17, 18, 21]). See [10] for detail.

We have $0 = 0_X$, i.e. $0_X(x) = 0$ for all $x \in X$, and $1 = \eta_X$. Further, for $f_1, f_2 \in Hom(X, \mathsf{FT}_\Omega X)$, we have

$$f_1 + f_2 = f_1 \vee f_2,$$

i.e. pointwise according to $(f_1 + f_2)(x) = f_1(x) \vee f_2(x)$, and

$$f_1 \cdot f_2 = f_1 \diamond f_2$$

where $f_1 \diamond f_2 = \mu_X \circ \mathsf{FT}_\Omega f_2 \circ f_1$ is the composition of morphisms in the corresponding Kleisli category of $\mathbf{F}$.

In [10] we also showed that $\mathbf{L} \bullet \mathbf{T}_\Omega$ can be extended to a partially ordered monad. In this case $\mathtt{Doc}_1 + \mathtt{Doc}_2$ means essentially sharpening the uncertainties, whereas $\mathtt{Doc}_1 \cdot \mathtt{Doc}_2$ is composition of information along a path of tasks.

Partially ordered monads have also been applied e.g. to rough sets [6], with substitutions having the interpretation of generalized relations.

6 Conclusions

We have provided some initial steps showing how syntactic BPMN can be extended with a semantics based on monad-oriented programming. The benefits of using Kleisli algebras of substitutions are methodologies for managing uncertainties as well as composing data and information delivered from paths of tasks and sub-processes.

The scenario including emergency medical care is very information and guideline intensive and serves well as a demonstrator for our approach to BPMN semantics. In future work we will further develop our methodology to include also monad based general logics as decision support mechanisms within sub-processes.

In extended versions of this paper and in future work we will expand our discussion on the case of emergency care including more detail and rich illustration in particular based on concrete information found in treatment guidelines. Further we will provide more detail on the advantages of using our proposed method. The formal part in this paper is very concise and needs to be expanded in particular for non-categorical readers.

Acknowledgement

We are grateful to the anonymous referee for helpful comments and suggestions.

References

[1] *Akutsjukvården i Västerbotten*, BEHANDLINGSRIKTLINJER, Västerbottens läns landsting, 2004, 2007.

[2] J. Adámek, H. Herrlich, G. Strecker, *Abstract and concrete categories*, Wiley-Interscience, New York, NY, USA, 1990.

[3] *Business Process Modeling Notation (BPMN)*, Version 1.2, January 2009, Object Management Group (OMG), 2008.

[4] P. Eklund, W. Gähler, *Fuzzy Filter Functors and Convergence*, Applications of category theory to fuzzy subsets (ed. S. E. Rodabaugh, E. P. Klement, U. Höhle), Theory and Decision Library B, Kluwer, 1992, 109-136.

[5] P. Eklund, W. Gähler, *Completions and Compactifications by Means of Monads*, in: Fuzzy Logic, State of the Art, R. Lowen and M. Roubens (eds.), Kluwer, 1993, 39-56.

[6] P. Eklund, M.A. Galán, J. Karlsson, *Categorical innovations for rough sets*, In: Rough Set Theory: A True Landmark in Data Analysis (Eds. A. Abraham, R. Falcon, R. Bello), Studies in Computational Intelligence, Vol. 174, Springer, 2009, 45-69.

[7] P. Eklund, M. A. Galán, J. Kortelainen, L. N. Stout, *Paradigms for non-classical substitutions*, Proc. 39th IEEE International Symposium on Multiple-Valued Logic (ISMVL 2009), May 21-23, 2009, Naha, Okinawa (Japan), 77-79.

[8] P. Eklund, M. A. Galán, M. Ojeda-Aciego, A. Valverde, *Set functors and generalised terms*, Proc. IPMU 2000, 8th Information Processing and Management of Uncertainty in Knowledge-Based Systems Conference, vol. III, 2000, 1595-1599.

[9] P. Eklund, M. A. Galán, J. Medina, M. Ojeda Aciego, A. Valverde, *Powersets of terms and composite monads*, Fuzzy Sets and Systems, **158** (2007), 2552-2574.

[10] P. Eklund, R. Helgesson, *Composing partially ordered monads*, 11th International Conference on Relational Methods in Computer Science, RelMiCS11, (Eds. R.Berghammer, A. Jaoua, B. Möller), Lecture Notes in Computer Science (2009), to appear.

[11] P. Eklund, F. Klawonn, Neural Fuzzy Logic Programming, FUZZ-IEEE '92, San Diego, March 8-12, 1992, *IEEE Trans. Neural Networks*, **3** No 5 (1992), 815-818.

[12] W. Gähler, *A topological approach to structure theory*, Math. Nachr. **100** (1981), 93-144.

[13] W. Gähler, *General Topology – The monadic case, examples, applications*, Acta Math. Hungar. **88** (2000), 279-290.

[14] W. Gähler, P. Eklund, *Extension structures and compactifications*, In: Categorical Methods in Algebra and Topology (CatMAT 2000), 181-205.

[15] J. A. Goguen, *L-fuzzy sets*, J. Math. Anal. Appl. **18** (1967), 145-174.

[16] S. C. Kleene, *Representation of events in nerve nets and finite automata*, In: Automata Studies (Eds. C. E. Shannon, J. McCarthy), Princeton University Press, 1956, 3-41.

[17] D. Kozen, *Kleene algebra with tests*, ACM Transactions on Programming Languages and Systems **19** (1999), 427443.

[18] W. Kuich, A. Salomaa, *Semirings, Automata, and Languages*, Springer-Verlag, berlin, 1986.

[19] E. Manes, *Algebraic Theories*, Springer, Berlin, 1976.

[20] J. Meseguer, *General logics*, In: Logic Colloquium '87 (Eds. H.-D. Ebbinghaus et al), Elsevier, North-Holland (1989), 275-329.

[21] A. Salomaa, *Two complete axiom systems for the algebra of regular events*, J. ACM **13** (1966), 158-169.