Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

# A Workflow Merging Approach for Emergency Procedures

*Rapporto Tecnico N.:*
**RT-ICAR-PA-16-01**

**Data:**

**febbraio 2016**

Consiglio Nazionale delle Ricerche, Istituto di Calcolo e Reti ad Alte Prestazioni (ICAR)
– Sede di Cosenza, Via P. Bucci 41C, 87036 Rende, Italy, URL: *www.icar.cnr.it*
– Sede di Napoli, Via P. Castellino 111, 80131 Napoli, URL: *www.na.icar.cnr.it*
– Sede di Palermo, Viale delle Scienze, 90128 Palermo, URL: *www.pa.icar.cnr.it*

Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

# A Workflow Merging Approach for Emergency Procedures

P. Ribino[1], C. Lodato[1] , M. Cossentino[1]

***Rapporto Tecnico N.:***
**RT-ICAR-PA-16-01**

***Data:***
**febbraio 2016**

[1] Istituto di Calcolo e Reti ad Alte Prestazioni, ICAR-CNR, Sede di Palermo, Viale delle Scienze edificio 11, 90128 Palermo.

*I rapporti tecnici dell'ICAR-CNR sono pubblicati dall'Istituto di Calcolo e Reti ad Alte Prestazioni del Consiglio Nazionale delle Ricerche. Tali rapporti, approntati sotto l'esclusiva responsabilità scientifica degli autori, descrivono attività di ricerca del personale e dei collaboratori dell'ICAR, in alcuni casi in un formato preliminare prima della pubblicazione definitiva in altra sede.*

# A Workflow Merging Approach for Emergency Procedures

P.Ribino, M.Cossentino, C.Lodato

*Istituto di Calcolo e Reti ad alte Prestazioni*
*Consiglio Nazionale delle Ricerche - Italy*

**Abstract**

Workflow merging is the problem to create a process model by unifying several process models that share process fragments. This paper presents a novel approach for merging emergency procedures to cope with the problem to manage multi emergencies. Such an approach is based on a new formalization of a process in terms of norms and goals. Norms are moreover used for solving conflict during the merging process. The proposed approach tries to overcome the limitations of design time merging approaches.

*Keywords:*

## 1. Introduction

Emergency management [1] is the discipline dealing with the creation of emergency plans through which institutions try to avoid potential risk and decrease the impact of disasters. The range of situations that could possibly involve emergency management is extensive. An emergency plan includes emergency procedures to address specific types of situations. An emergency procedure (also named protocol) is a set of actions (i.e: a process) to be conducted in a certain order when some situations occur. Commonly, these procedures are defined to deal with a single event. Hence, procedures for addressing terrorism, industrial accidents, natural calamity (such as earthquakes, fires, landslides) are ad hoc designed. When multi emergencies and unpredicted scenarios simultaneously occur, a real time fusion of emergency procedures along with the adoption of appropriate regulations is mandatory in order to avoid waste of resources.

This kind of issue can be compared with the problem of workflow merging in the context of Process Management. Process management is a field of combining management and technology focused on aligning organizations with the needs of clients. The basic element of process management is the Business Process. A business process is a set of one or more linked procedures or activities, which are collectively executed to reach required business objectives normally within the context of an organizational structure defining functional roles and relationships.

In such context, it often occurs that several companies must make frequent business process changes as well as organizational changes due to mergers and acquisitions. Hence, multiple alternative processes, previously belonging to different companies, need to be consolidated into a single one in order to eliminate redundancies and create synergies. To this end, teams of business analysts need to compare similar process models so as to identify commonalities and differences, and to create integrated process models. Combining different procedures is a time-consuming and error-prone task. In the context of business organizations few manual or semi-automatic methods are proposed[2, 3]. Such methods work at design time. No timely solutions are required.

In the context of multi-emergency, where several unexpected situations may occur due to the combined effect of multiple events, the number of possible paths the combined process has to take into consideration may dramatically increase. The approaches proposed in the field of business processes are not properly adequate. They could be used at design time for creating multi-emergency procedures, but always with the limitation of the unpredictability of the situations. In the context of emergency management, well-timed solutions are crucial.

In this work we propose a method that allows a flexible run-time process merging by including normative concepts in a goal-based paradigm for modelling processes. In our approach, norms and goals allow to cope with two main aspects of a process. Goals expresses what a process has to reach, what is the desired state of the world a process has to result in. Norms denote the way a process has to be conducted in order to achieve the desired state of the world in compliance with the normative context in which that process takes place. *Norms* represent process regulations by specifying obligations, permissions or prohibitions to be followed during process activities in case of certain conditions occur, thus relaxing or restricting a process.

The main contribution of this paper is a method for merging multiple procedures in a real time fashion. For achieving our purpose, we propose a merging algorithm that generates a single merged process starting from several processes. Such an algorithm is based on a novel modelling approach that allows to represent a process in terms of norms and goals. Hence, the merged process could be executed by goal-oriented workflow engine. In this paper, we tested our approach by developing a web application that uses MUSA[4] as a workflow management engine for real time execution of emergency procedures.

We use BPMN[5] and SBVR[6] as standard notations in order to show the input and output of merging algorithms.

The research has been partially funded by SIGMA project -*Sistema Integrato di sensori in ambiente cloud per la Gestione Multirischio Avanzata*. SIGMA is a national research project developed in collaboration with several research and industrial partners, its aim is the development of an integrated framework for multi-emergency management.

The rest of the paper is organized as follows.

## 2. Theoretical Background

An emergency procedure is a set of actions (i.e: a process) to be conducted in a certain order when some situations occur in order to reach a desired result, that is addressing the emergency. Commonly, it is modelled by using a BPMN notation in order to graphically represent the workflow.

The aim of this section is to provide the theoretical background the research presented in this paper is based on. This section is organized in two parts: the first one introduces the modeling language used for representing emergency procedure and its regulations. The second one provides an overview of the adaptive framework for runt-time modification of workflows in which we have implemented our proposed method for workflow fusion.

### *2.1. Modeling Business Processes: BPMN & SBVR*

Business Process Modelling Notation (BPMN)[5] and the Semantic Business Vocaboulary Rules (SBVR) [6] are widely recognized and well know standards that allow to model each aspect of a process.

In particular, BPMN is a graph-oriented notation developed by Object Management Group (OMG) that was conceived for being highly understandable by all business people (i.e: business analysts, technical developers and business users) interested into business processes. Practically, BPMN consists of a set of graphical objects can be connected almost arbitrarily. Such graphical elements are grouped into four basic categories: *Flow Objects*, *Connecting Objects*, *Swimlanes* and *Artifacts*.

*Flow Objects* refer to elements that are connected together to form a complete process flow. In a BPMN diagram it is possible to distinguish the following Flow Objects: *Event* that represents something that happens during the course of a business process; *Activity* that is a generic work that company performs; and *Gateway* that is used for controlling how a business process flows. Such objects are connected together in a diagram to create the basic structure of a business process by means of *Connecting Objects*. There are three possible connecting objects: *Sequence Flow* is used to show the order (the sequence) that activities will be performed in a Business Process; *Message Flow* is used to show the flow of messages between two separate business participants that send and receive them; and *Association* is used to associate data, text, and other Artifacts with flow objects. As concerns the *Swimlanes*, they represent a mechanism to organize activities in order to illustrate different responsibilities and process participants. Finally, *Artifacts* are notational element for adding additional information to the business process. They are: *Data Objects* that allow to show how data is required or produced by activities; *Group* is a mean used for grouping element for analysis purposes and finally *Annotations* can be used to provide additional text information for the reader of a BPMN Diagram.

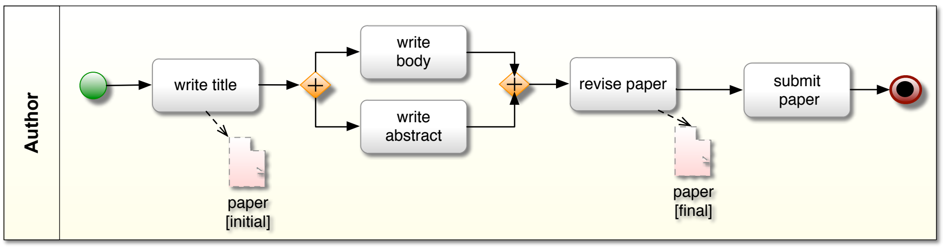


Figure 1: An example of a BPMN model of a Business Process.

Figure 1 shows a simple BPMN diagram that models a process for writing a paper to be submitted to a conference by using the most common notational elements such as activitie (e.g: *write title*, *revise paper*, etc...), pools (e.g: *author*) and data objects (e.g: *paper*).

Whilst BPMN models the dynamics of business processes, SBVR [6] allows to model other complementary aspects of a business process such as in particular business constraints. Generally, a business constraint describes the conditions of a business process execution. In the business context the term "constraint" commonly refers to the broader understanding of so called business rules (BR). A business rules may define the semantics of business concepts, reactions to business events, constraints and preconditions on tasks and activities, as well as the prohibitions, permissions and obligations of business actors and activities. In other words, business rules guide and constrain various aspects of business, including the sequence and timing of activities [7]. Thus, SBVR allows to model business vocabularies construction and business rules definitions (elements of guidance that govern actions). However, SBVR does not standardize any particular language for expressing vocabularies and rules. Instead, SBVR uses 'semantic formulation', which is a way of describing the semantic structure of statements and definitions. This approach of specifying structures of meaning, with its sound theoretical foundation of formal logic, provides a formal, language-independent means for capturing the semantics of a community's body of shared meanings. SBVR separates semantic formulations from meanings. Meaning is divided into two categories: *Concepts*that are classifiers of things (noun concepts) and classifiers of states and actions (verb concepts or fact types) and *Propositions*that are meanings of statements (rules also called element of guidance). SBVR further divides the meaning of rule into the following subcategories:

- Structural or definitional rules: They are used to define an organizations setup. They are always true and cannot be violated (necessity, possibility).
- Behavioural or operational rules: They express the conduct of an entity and can be violated (obligation, prohibition).

SBVR includes constructs called semantic formulations that structure the meaning of rules or the definition of concepts. There are two kinds of semantic formulations: logical formulations and projections. Logical formulations include modal operators, logical operations, quantifications, atomic formulations based on fact types. Projections are used to formulate definition of meaning and also impose constraints or restriction on concepts. SBVR also proposes a Structured English (SSE) which is one of the multiple concrete syntax which can be used to designate textually the statements with formal meaning that could be mapped to SBVR concepts. SSE four formatting style as follows:

- terms: underlined text is used for representing noun concepts defined in the vocabulary (e.g: author );
- names: double-underlined green text is used for represented proper nouns (e.g: Italy);
- verbs: blue italic text is used for representing fact types (e.g: author is member of academic institution);
- keywords: red text is used for representing linguistic symbols used to construct statements and definitions (e.g: It is necessary that)

For example, we can define by using the SSE notation a structural rule for the process shown in Figure 1 as follows:

It is necessary that an author is member of an academic institution.
Analogously, a behavioural rule could be:

It is prohibited that an author submits a paper after deadline.

### *2.2. Middleware for User-driven Service Adaptation*

The work proposed in this paper extends the conceptual framework proposed in [4] by adding normative concepts and workflow merging methods.

The Middleware for User-driven Service Adaptation (MUSA) proposed in [4] is intended to provide means for supporting run-time adaptation of a process together with a multi agent system for executing the activities of the process.

The core of such framework lies on the Proactive Means-End Reasoning where its theoretical foundation are based on three key concepts:

- *state of the world* that is the knowledge (i.e: a set of beliefs) about the world in a given moment the system owns;
- *goal* that is a desired change in the state of the world a user wants to achieve;
- *capability* that is a run-time property of the system that may be intentionally used to address a given result. The effect of a capability is an endogenous evolution of the state of the world.

The algorithm that implements the Proactive Means-End Reasoning [8] is able to solve the following problem:

*Given a current state of the world $W_t$, a Goal Model*[1] *(G, R) and a set of available Capabilities $C$, the Proactive Means-End Reasoning concerns finding a set of capabilities $CS \subseteq C$ in which each capability will address one of the goals of the Goal Model (G,R), thus to grant the achievement of the root goal.*

The MUSA is also endowed with a module (i.e: *GoalSpec Translator*) able to translate BPMN business process in a set of goals expressed in GoalSPEC language [9]. The GoalSPEC language has been specifically designed for enabling runtime goal injection into the system and software agent reasoning. A goal in GoalSPEC is composed of *Trigger Condition* and *Final State.* The trigger condition is an event that must occur in order to start acting for addressing the goal. The final state is the desired state of the world that must be addressed. For example, looking at Figure 1, the GoalSpec Translator

[1]A goal model is a directed graph, (G,R) where G is a set of goals (nodes) and R is the set of Refinement and Influence relationships (edges). In a goal model there is exactly one root goal, and there are no refinement cycles.

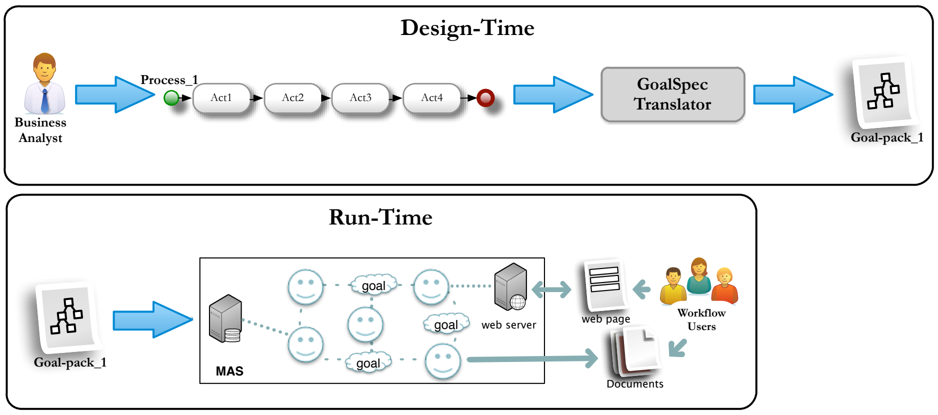


Figure 2: A functional schema of the workflow engine

converts the activity *revise paper* in the following goal:

WHEN *done(write_body)* AND *done(write_abstract)* THE *author* SHALL PRODUCE *done(revise_paper)* AND *final(paper)*

where the Trigger Condition is [*done(write_body)* AND *done(write_abstract)*] and the Final State is [*done(revise_paper)* AND *final(paper)*].

For the sake of clarity in Figure 2 is reported a possible usage scenario (the most simple one) of the framework[2]. In such scenario, as common a business analyst models a business process as BPMN workflow. The GoalSpec Translator converts the BPMN process in a goal pack, that is a set of Goal-SPEC goal. This resulting goal pack is injected at Run-Time in the multi agent system that executes the workflow adopting the proactive means-end reasoning in order to decide how to address an injected goal. The agent that owns the capabilities to achieve a goal provides the users the workflow users the appropriate resources (i.e: web pages, documents, services and so on) to accomplish the activities of the BPMN process.

In this paper we propose an extension of the framework that make possible

[2]It is out of the paper to discuss the self-adaptation of the framework and the run-time goal injection. Our purpose is to introduce the fundamental concepts the reader needs in order to understand the proposed method.

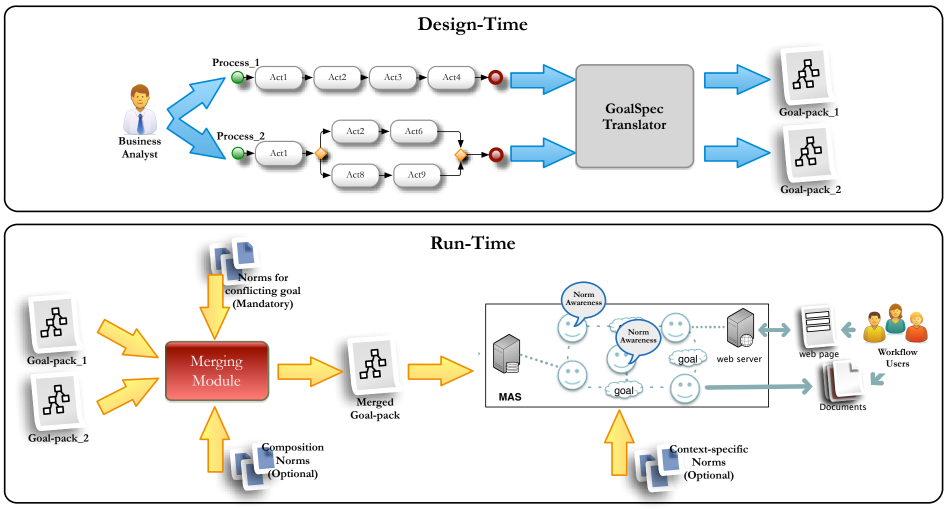


Figure 3: A functional schema of the workflow engine with merging

the usage scenario shown in Figure 3.

It deals with two kinds of goal: social goal and individual goal. Social goals are collective goals that specify top-level goals which achievement may be obtained by addressing lower-level social goals or individual goals. Individual goals are related to generating a specific outcome in the workflow. In the next section we present the theoretical basis of our work.

## 3. Norms and Goals for modelling processes

The process merging approach proposed in this paper is build on some theoretical foundations we introduce in this section. In order to exemplify, we refer to a simple process for submitting a paper. Figure 1 shows a BPMN diagram that models such process by using the most common notational elements such as activities (e.g: *write title*, *revise paper*, etc...), pools (e.g: *author*) and data objects (e.g: *paper*).

The basis of our approach lies on a process specification in terms of goals and norms, as follows:

▼ Definition 3.1 — *Process Specification*

A Process Specification is defined by the elements of the following triple:

$$\langle \mathcal{G}, \mathcal{R}, \mathcal{N} \rangle$$

where

- $\mathcal{G}$ is a set of *Goals* that are states of the world the actors of the process wants to achieve;
- $\mathcal{R}$ is a set of *Roles* played by the actors of the process for achieving goals.
- $\mathcal{N}$ is a set of norms that constraint the admissible states of the world.

Before to formally define goal and norm, it is useful to introduce a further key concept of the approach: *the state of the world.* The state of the world represents a set of declarative information concerning particular conditions or set of circumstances in which the process operates in at specific time. In this paper we assume that goals, norms and state of the world refer to the same knowledge domain. It is out of the scope of the paper how to model this knowledge. We assume that a knowledge domain is represented by means of a predefined set of concepts and predicate.

▼ Definition 3.2 — *State of the world*

Let $\mathcal{D}$ the set of concepts of a knowledge domain. Let $\mathcal{L}$ be a first-order logic defined on $\mathcal{D}$ with $\top$ a tautology and $\bot$ a logical contradiction, where an atomic formula $p(t_1, t_2..., t_n) \in \mathcal{L}$ is represented by a predicate applied to a tuple of terms $(t_1, t_2..., t_n) \in \mathcal{D}$ and the predicate is a property of or relation between such terms that can be true or false.

A *state of the world* in a given time t ($\mathcal{W}^t$) is a subset of atomic formulae whose values are true at the time t:

$$\mathcal{W}^t = [p_1(t_1, t_2, ..., t_h), ..., p_n(t_1, t_2, ..., t_m)]$$

For example, let $D = \{paper, author, deadline\}$ a set of concepts related to a knowledge domain defined for the paper submission process (see Figure 1). A possible state of the world at time t could be expressed by the following atomic formulae: $\mathcal{W}^t = [before(deadline), done(submit\_paper)]$. This means that at time t to which the state of the world refers, the time for paper submission is not expired yet and the author has submitted the paper. Definition 3.2 is based on close world hypothesis that assumes all facts that are not in the state of world are considered false.

The second element of process specification is the concept of goal. It is widely used in software engineering and artificial intelligence to implement sophisticated reasoning mechanism. Goal is defined as "a state of affair an actor wants to achieve" [10]. For the scope of this paper we use a revised version of the formal definition of goal taken from [8].

▼ DEFINITION 3.3 — *Goal*

Let $\mathcal{D}$, $\mathcal{L}$ and $p(t_1, t_2..., t_n) \in \mathcal{L}$ as previously introduced in definition 3.2. Let $t_c \in \mathcal{L}$ and $f_s \in \mathcal{L}$ formulae that may be composed of atomic formulae by means of logic connectives AND($\wedge$), OR ($\vee$) and NOT ($\neg$).

A *Goal*is a pair $\langle t_c, f_s \rangle$ where $t_c$ (*trigger condition*) is a condition to evaluate over a state of the world $\mathcal{W}^t$ when the goal may be actively pursued and $f_s$ (*final state*) is a condition to evaluate over a state of the world $W^{t+\Delta t}$ when it is eventually addressed:

$$\textit{a goal is active iff } t_c(\mathcal{W}^t) \wedge \neg f_s(\mathcal{W}^t) = true$$
$$\textit{a goal is addressed iff } f_s(\mathcal{W}^{t+\Delta t}) = true$$

Moreover, llet two goals $g_1 = \langle t_{c1}, f_{s1} \rangle$ and $g_2 = \langle t_{c2}, f_{s2} \rangle$:

- the goal $g_1$ *is equivalent to* ($\simeq$) $g_2$ if the final state $f_{s1}$ is logically equivalent[3] ($\Leftrightarrow$) to $f_{s2}$,

$$g_1 \simeq g_2 \text{ iff } f_{s1} \Leftrightarrow f_{s2}.$$

- the goal $g_1$ *depends on* ($\to$) $g_2$ if the trigger condition $t_{c1}$ implies the final state $f_{s2}$ or if the final state $f_{s_1}$ implies the final state $f_{s2}$,

$$g_1 \to g_2 \text{ iff } t_{c1}(\mathcal{W}^t) \Rightarrow f_{s2}(\mathcal{W}^t) \vee f_{s_1}(\mathcal{W}^t) \Rightarrow f_{s2}(\mathcal{W}^t)$$

With respect to the previous example a possible goal could be $g = \langle done(write\_paper), done(submit\_paper)\rangle$. This means an author can achieve the desired state of affairs, namely the submission of his own manuscript, only when the paper has been written. In other words, $g$ is active when the state of the world in a given time t is $\mathcal{W}^t = [\ldots, done(write\_paper), \ldots]$. Conversely, $g$ is addressed only when exist a state of the world in a given time $t + \Delta t$ in which $done(submit\_paper)$ is true, for example $\mathcal{W}^{t+\Delta t} = [done(write\_paper), before(deadline), done(submit\_paper)]$.

Moreover, the equivalence relationship between two goals $g_1$ and $g_2$ implies that if $g_1$ is addressed in a state of the world also $g_2$ is addressed in the same state of the world. This is because the equivalence relationship among goal lies on the logical equivalence between their final states.

Finally, the key element of the process specification is the concept of norm. We define norms in such away they can be used for constraining the admissible state of the world by means of permissions, obligations or prohibitions.

▼ Definition 3.4 — *Norm*

Let $\mathcal{D}$, $\mathcal{L}$ and $p(t_1, t_2 ..., t_n) \in \mathcal{L}$ as previously introduced in definition 3.2. Let $\phi \in \mathcal{L}$ and $\rho \in \mathcal{L}$ formulae composed of atomic formula by means of logic connectives AND($\wedge$), OR ($\vee$) and NOT ($\neg$).
Moreover, let $D_{op} = \{permission, obligation, prohibition\}$ the set of deontic operators.

[3]In logic, two statements are logically equivalent if they have the same truth value in every model.

A *Norm* is defined by the elements of the following tuple:

$$n = \langle r, g, \rho, \phi, d \rangle$$

where

- $r \in \mathcal{R}$ is the *Role* the norm refers to. The special character "_" indicates that the norms refers any role.
- $g \in \mathcal{G}$ is the *Goal* the norm refers to.The special character "_" indicates that the norms refers to any goal.
- $\rho \in \mathcal{L}$ is a formula expressing the set of actions and state of affairs that the norm disciplines.
- $\phi \in \mathcal{L}$ is a logic condition (to evaluate over a state of the world $\mathcal{W}^t$) under which the norm is applicable;
- $d \in D_{op}$ is the deontic operator applied to $\rho$ that the norm prescribes to the couple $(r, g) \in \mathcal{R} \times \mathcal{G}$.

  In particular $d(\rho) = \begin{cases} \rho & \text{iff } d = \textit{obligation} \\ \neg\rho, & \text{iff } d = \textit{prohibition} \\ \rho \vee \neg\rho & \text{iff } d = \textit{permission} \end{cases}$

In other words, let a state of the world $\mathcal{W}^t$ a norm prescribes to a couple $(r, g)$ the deontic operator $d$ applied to $\rho$ if $\phi$ is true in $\mathcal{W}^t$.

With respect to the previous example a norm could be

$$n = \langle \_, \_, done(submit\_paper), after(deadline), prohibition \rangle$$

which prescribes to anyone the prohibition to submit a paper (i.e: $d(\rho) = \neg done(submit\_paper)$) after the deadline.

In the following we introduce two further definitions that characterize the concept of state of the world and norm. Such definitions allow to implement sophisticated reasoning in the proposed process merging algorithms. In particular we define: *(i)* the concept of *Inadmissible State of the World* in order to determine the boundary within which a process has to be carried out; and *(ii)* the concept of *State of Norm* for determining the particular condition that a norm is in at a specific time.

▼ Definition 3.5 — *Inadmissible State of the World*

A state of the world at a given time t

$$\mathcal{W}^t = [p_1(t_1, t_2, ..., t_h), ..., p_n(t_1, t_2, ..., t_m)]$$

is an *Inadmissible State of the World* iff $\exists\ n = \langle r, g, \rho, \phi, d \rangle \mid$

$$\begin{cases} \phi(\mathcal{W}^{t-\Delta t}) \wedge \neg\rho(\mathcal{W}^{t-\Delta t}) = true \\ p_1(t_1, t_2, ..., t_h) \wedge ... \wedge p_n(t_1, t_2, ..., t_m) \wedge d(\rho) = \perp \end{cases}$$

It is worth noting that $\phi$ and $\rho$ have to be evaluate in $\mathcal{W}^{t-\Delta t}$. This means that a state of the world at a given time t is an Inadmissible State of the world if at time $t - \Delta t$ the condition under which the norm is applicable is true and the process is not in the state of affair the norm disciplines.

In order to exemplify, let us consider the previous norm that prohibits the submission of a paper after the deadline

$$n = \langle \_, \_, done(submit\_paper), after(deadline), prohibition \rangle$$

Let assume that at time $t$ the state of the world is

$$\mathcal{W}^t = [done(write\_paper), after(deadline)]$$

Then let us suppose that in some way someone or something has changed the state of the world and at time $t + \Delta t$

$$\mathcal{W}^{t+\Delta t} = [done(write\_paper), after(deadline), done(submit\_paper)]$$

According to the previous definition $\mathcal{W}^{t+\Delta t}$ is an Inadmissible State of the world because:

$$\underbrace{after(deadline)}_{\phi(\mathcal{W}^t)} \wedge \underbrace{\neg done(submit\_paper)}_{\neg\rho(\mathcal{W}^t)} = true$$

$$\underbrace{done(write\_paper) \wedge after(deadline) \wedge done(submit\_paper)}_{\mathcal{W}^{t+\Delta t}} \wedge$$
$$\wedge \underbrace{\neg done(submit\_paper)}_{d(\rho)} = \perp$$

Indeed, in $\mathcal{W}^t$ the predicate $done(submit\_paper)$ is false for the close world hypothesis. Thus $\neg done(submit\_paper)$ is true.
In $\mathcal{W}^{t+\Delta t}$, $done(submit\_paper)$ is true by definition (see definition 3.2). Moreover, in this case $d(\rho)$ is equal to $\neg done(submit\_paper)$. This makes the second condition of definition 3.5 a logical contradiction.

In order to clarify, we show another example considering an obligation norm. Let us suppose a norm that obligates the submission of the camera ready if the paper was accepted.

$$n = \langle \_, \_, \underbrace{done(submit\_camera\_ready)}_{\rho},$$
$$\underbrace{accepted(paper) \wedge before(camera\_ready\_deadline)}_{\phi}, obligation\rangle$$

Let assume that at time $t$ the state of the world is:

$$\mathcal{W}^t = [done(write\_paper), accepted(paper), before(camera\_ready\_deadline)]$$

Then let us suppose at time $t + \Delta t$ nothing is changed in the state of the world except the time:

$$\mathcal{W}^{t+\Delta t} = [done(write\_paper), accepted(paper), after(camera\_ready\_deadline)]$$

According to the previous definition $\mathcal{W}^{t+\Delta t}$ is an Inadmissible State of the World because:

$$\underbrace{accepted(paper) \wedge before(camera\_ready\_deadline)}_{\phi(\mathcal{W}^t)} \wedge \underbrace{\neg done(submit\_camera\_ready)}_{\neg\rho(\mathcal{W}^t)} = true$$

$$\underbrace{done(write\_paper) \wedge accepted(paper) \wedge after(camera\_ready\_deadline)}_{\mathcal{W}^{t+\Delta t}} \wedge$$

$$\wedge \underbrace{done(submit\_camera\_ready)}_{d(\rho)} = \bot$$

Indeed, in $\mathcal{W}^t$ the predicate $done(submit\_camera\_ready)$ is false for the analogous considerations of the previous example.

In $\mathcal{W}^{t+\Delta t}$, $done(submit\_camera\_ready)$ is still false. This is equivalent to have in the $\mathcal{W}^{t+\Delta t}$ the following formula $\neg done(submit\_camera\_ready)$. As a consequence, $\mathcal{W}^{t+\Delta t}$ the same predicate has at the same time value both false and true. This makes the second condition a logical contradiction.

Conversely, let us consider the previous norm

$$n = \langle \_, \_, done(submit\_paper), after(deadline), prohibition \rangle$$

Let assume that at time $t$ the state of the world is

$$\mathcal{W}^t = [done(write\_paper), after(deadline), done(submit\_paper)]$$

Then let us suppose at time $t + \Delta t$ that nothing is changed in the state of the world except the time:

$$\mathcal{W}^{t+\Delta t} = [done(write\_paper), after(deadline), done(submit\_paper)]$$

In this case although the second condition of definition 3.5 is the same of the first case:

$$\underbrace{done(write\_paper) \wedge after(deadline) \wedge done(submit\_paper)}_{\mathcal{W}^{t+\Delta t}} \wedge$$

$$\wedge \underbrace{\neg done(submit\_paper)}_{d(\rho)} = \bot$$

The first one is not satisfied:

$$\underbrace{after(deadline)}_{\phi(\mathcal{W}^t)} \wedge \underbrace{\neg done(submit\_paper)}_{\neg\rho(\mathcal{W}^t)} = false$$

The evaluation of the first condition allows to avoid to consider inadmissible a state of the world that a process reached in a given moment in which the condition of the norm was not valid.

Hence, it results useful introduce also the concept of State of Norm.

▼ DEFINITION 3.6 — *State of Norm*

Let a norm $n = \langle r, g, \rho, \phi, d \rangle$ where $g = \langle t_c, f_s \rangle$ and let a state of the world in a given time t ($\mathcal{W}^t$)

A norm can assume the following states:

- $n$ is *applicable at time t* iff $\phi(\mathcal{W}^t) = true \vee \phi = \top$
- $n$ is *active at time t* iff n is applicable and $t_c(\mathcal{W}^t) = true$
- $n$ is *logically contradictory* iff $\phi\ is\ \bot$
- $n$ is *violated* at time t iff $\mathcal{W}^t$ is an inadmissible state of the world
- $n$ is *in opposition to goal* iff $f_s \wedge d(\rho)\ is\ \bot$
- $n$ is *incompatible with goal* iff it is applicable and $\nexists\ \mathcal{W}^t \mid f_s(\mathcal{W}^t) = true$[4].

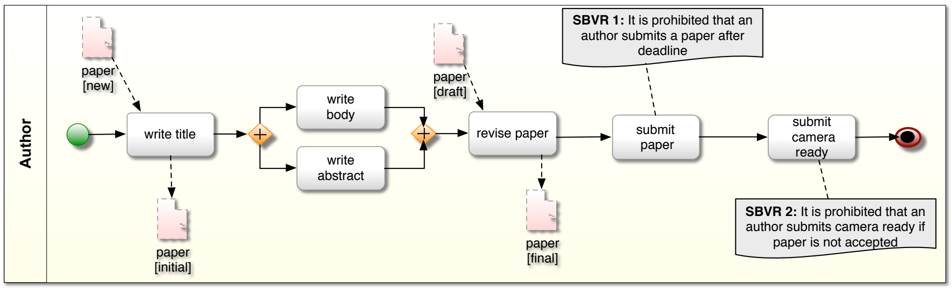
Moreover, let a state of the world in a given time t ($W^t$) and let two norms $n_1 = \langle r_1, g_1, \rho_1, \phi_1, d_1 \rangle$ and $n_2 = \langle r_2, g_2, \rho_2, \phi_2, d_2 \rangle$ where $r_1 = r_2,\ g_1 = g_2$ $\rho_1 = \rho_2$

- $n_1$ and $n_2$ are *deontically contradictory* iff $\begin{cases} \phi_1(\mathcal{W}^t) \wedge \phi_2(\mathcal{W}^t) = true \\ d_1 \neq d_2 \end{cases}$

It is worth noting that we talk about *logically contradictory* when the contradiction concerns the logical conditions ($\phi \in \mathcal{L}$) under which the norms are applicable. At the contrary, we talk about *deontically contradictory* when the contradiction concerns the semantic meaning of the deontic operator ($d \in D_{op}$) the norms apply.

In order to simplify the concepts previously introduced, let us consider the BPMN workflow depicted in Figure 4 that models a business process for

[4]A trivial example of this situation could be a norm prescribing that is always prohibited to bathe applied to the goal "swimming". In this case the norm is applicable in any state of the world because there are no condition to be verified but the fulfilment of the goal leads always in a not admissible state of the world because it is impossible to swim without to bathe.



Author
paper [new]
write title
paper [initial]
write body
write abstract
paper [draft]
revise paper
paper [final]
SBVR 1: It is prohibited that an author submits a paper after deadline
submit paper
submit camera ready
SBVR 2: It is prohibited that an author submits camera ready if paper is not accepted

Figure 4: A BMPN/SBVR model of a workflow for submitting a paper.

submitting a paper. Let us consider that an extract of the set of concepts defining the business domain is:

$$\mathcal{D} = \{paper, author, deadline, title, abstract, write\_abstract, submit\_paper, \ldots\}$$

According to the definition 3.1 the process shown in Figure 4 can be expressed as follows:

$$\mathcal{G} = \left\{ \begin{array}{l} goal_{social} = \langle new(paper), [final(paper) \wedge done(revise\_paper)] \rangle \\ g_1 = \langle new(paper), [initial(paper) \wedge done(write\_title)] \rangle \\ g_2 = \langle [initial(paper) \wedge done(write\_title)], done(write\_body) \rangle \\ g_3 = \langle [initial(paper) \wedge done(write\_title)], done(write\_abstract) \rangle \\ g_4 = \langle [draft(paper) \wedge done(write\_title) \wedge done(write\_abstract)], \\ \quad\quad [final(paper) \wedge done(revise\_paper)] \rangle \\ g_5 = \langle [final(paper) \wedge done(revise\_paper)], done(submit\_paper) \rangle \\ g_6 = \langle done(submit\_paper), done(submit\_camera\_ready) \rangle \end{array} \right\}$$

The set of roles is:

$$\mathcal{R} = \{author\}$$

The set of norms is:

$$\mathcal{N} = \left\{ \begin{array}{l} n_1 = \langle author, g_5, done(submit\_paper), after(deadline), prohibition \rangle \\ n_2 = \langle author, g_6, done(submit\_camera\_ready), \neg accepted(paper), prohibition \rangle \end{array} \right\}$$

Let us to suppose that at a given time t

$$\mathcal{W}^t = \{done(revise\_paper)\}$$

The norm $n_1$ is not applicable because $\phi_1 = \mathit{after}(\mathit{deadline})$ is false in $\mathcal{W}^t$ whilst the norm $n_2$ is applicable because $\phi_2 = \neg \mathit{accepted}(\mathit{paper})$ is true in $\mathcal{W}^t$[5].

Let us to suppose that at a given time $t + \Delta t$ the state of the world is the following:

$$\mathcal{W}^{t+\Delta t} = \big\{\mathit{done}(\mathit{revise\_paper}), \mathit{after}(\mathit{deadline}), \mathit{accepted}(\mathit{paper})\big\}$$

In this case, the norm $n_1$ is applicable because $\phi_1 = \mathit{after}(\mathit{deadline})$ is true in $\mathcal{W}^{t+\Delta t}$ whilst the norm $n_2$ is not applicable because $\phi_2 = \neg \mathit{accepted}(\mathit{paper})$ is false in $\mathcal{W}^{t+\Delta t}$.

Finally, let two norms $n_1, n_2 \in \mathcal{N}$ where

$$n_1 = \langle \mathit{author}, g_6, \mathit{done}(\mathit{submit\_camera\_ready}), \mathit{accepted}(\mathit{paper}), \mathit{obligation}\rangle$$
$$n_2 = \langle \mathit{author}, g_6, \mathit{done}(\mathit{submit\_camera\_ready}), \mathit{after}(\mathit{deadline}), \mathit{prohibition}\rangle$$

Let us to suppose that the state of the world in a given time t ($W^t$) is

$$\mathcal{W}^t = \big\{\mathit{after}(\mathit{deadline}), \mathit{accepted}(\mathit{paper})\big\}$$

In such case both $\phi_1$ and $\phi_2$ are true in $\mathcal{W}^t$ making deontologically contradictory $n_1$ and $n_2$, according to definition 3.6.

In the following section, the algorithms for merging workflow are presented.

## 4. Algorithms for merging emergency procedures

We define workflow-merging as the process of combining several workflows into another by unifying redundant activities. We refer to the original workflows as *merging workflows* and the resulting workflow as the *merged workflow*.

In our approach *merging workflows* (i.e: emergency procedures) are initially specified according to BPMN/SBVR notation in order to maintain a standard representation that is understandable by the domain expert. Conversely, the *merged workflow* (i.e: multi-emergency procedure) is specified according to definition 3.1. It is out the scope of this paper the algorithm that convert the merged workflow into a BPMN workflow. For the sake of simplicity, we shows the *merged workflow* both according our specification and BPMN.

In our approach, norms can be context-specific norms that represent regulations of the normative environment the *merging workflows* have to obey. At the same time, norms can be appositely defined for avoiding process deadlock in the *merged workflow*.

[5]It is worth noting that all facts that are not in the state of world are considered false.

**Algorithm 1:** Workflow-merge algorithm

```
Data: $n$ standard (BPMN/SBVR) workflow models, $n \geq 2$
Result: Merged Workflow $WF_{merged} = \langle \mathcal{G}_{merged}, \mathcal{R}, \mathcal{N} \rangle$
// Loop for generating $WF_i = \langle \mathcal{G}_i, \mathcal{R}_i, \mathcal{N}_i \rangle$
① for $i \leftarrow 1$ to $n$ do
    $\mathcal{G}_i \leftarrow \varnothing$;
    ②foreach BPMN activity $a_j$ of $WF_i$ do
        translate $a_j$ into goal $g_j$;
        add $g_j$ to $\mathcal{G}_i$
    $\mathcal{N}_i \leftarrow \varnothing$;
    ③ foreach SBVR rule $r_h$ of $WF_i$ do
        translate $r_h$ into norm $n_h$;
        add $n_h$ to $\mathcal{N}_i$
    add $\mathcal{R}_i$ to $\mathcal{R}$
④ $\mathcal{G}_{Temp} \leftarrow goal\_fusion$ // see algorithm 2
⑤ if $\exists\ WF_i \mid \mathcal{N}_i \neq \varnothing, i = 1...n$ then
    $\mathcal{G}_{merged} \leftarrow combine\_norm\_and\_goal$ // see algorithm 3
else
    $\mathcal{G}_{merged} \leftarrow \mathcal{G}_{Temp}$
⑥ foreach $g \in G_{merged}$ do
    // see Algorithm 5
    if $isPursuable(g) = false$ then
        select norms related to g;
        revise norms;
```

The workflow-merge algorithm (see Algorithm 1) takes in input several *merging workflows* and produces an unique merged workflow. Some steps of the algorithm are grouped according to their function and have been separately presented.

In particular, step ① is a mandatory procedure for translating standards workflows according to the proposed process specification. In particular steps ② and ③ convert BPMN activities and SBVR rules respectively into goal and norms. For space concerns, we avoid to detail the algorithms that allow such conversion. They are basically two parsers that convert XML schema of BPMN/SBVR models into goals and norms specification according to definitions 3.3 and 3.4. For the purpose of this work, it is only worth nothing that this conversion produces two kinds of goal: *social goal* and *individual goal*. Social goals are collective goals that specify top-level goals which achievement may be obtained by addressing lower-level social goals or individual goals. Individual goals are related to generating a specific outcome in the workflow.

The set of resulting goals thus are initially combined (step ④) according to

Algorithm 2. Step ⑤ allows to encapsulate the condition expressed by the norms inside the goal they refer (see Algorithm 3). Then the whole fusion process is resulting in a new set of goals, namely $G_{merged}$. The final step of the algorithm makes a control about the satisfiability of each goal of $G_{merged}$.

Algorithm 2 shows how to merge goals. The input are the *merging workflows* expressed according to our specification (i.e: WF=$\langle\mathcal{G},\mathcal{R},\mathcal{N}\rangle$). It produces a set of goals, where *equivalent goals* (if any) are combined in a unique goal. Step ① allows for separately grouping social goals and individual goals. Social goals play a fundamental role in the goal fusion process. They contains global information about an entire *merging workflow*. Their fusion produces the social goal for the whole *merged workflow* (see Step ②). The key element of the goal fusion algorithm lies on the equivalence between goals (see Step ③). Two goals can be fused if they are equivalent goals (see Definition 3.3). The output of Algorithm 2 is a set of goals where some of them have been unified. Figure 5 shows a a graphical representation of Algorithm 2.

A further step for completing the whole merging process is to reconcile norms with goal they refer and above all with merged goal resulting from algorithm 2. This task is addressed by Algorithm 3. It allows to encapsulate the condition expressed by the norms inside the goal they refer. Encapsulating a norm condition into a goal modifies the activation of that goal making it compliant with norm. This algorithm consists in an initial pre-filtering of *logically contradictory norms* (see Step ①). Then norms are encapsulated into goals (see Step ②) by compositing new trigger conditions for goals from the norm conditions. Such composition (see Algorithm 4) takes into consideration different types of norm and addresses to the following question "when norms regulate a goal, in what cases that goal is activated?". The answer is: *i)* when the trigger condition is true and the norm is not applicable or *ii)* when the norm is applicable and its deontic operator is *permission*; *iii)* when the norm is active and its deontic operator is an *obligation*. Figure 6 shows the activation table of a goal regulated by a norm.

Finally, Algorithm 5 evaluates if exists at least one possible state of the world in which the goal may be actively pursued. Otherwise, it shows a list of norms related to the goal in order to revise them for ensuring goal satisfaction. This algorithm detects logical contradictions inside a merged goal thus ensuring the consistency of the *merged workflow*.

In the following section, some typical merging scenarios are presented.

**Algorithm 2:** Goal Fusion

```
Data: $n$ workflow models WF=$\langle\mathcal{G},\mathcal{R},\mathcal{N}\rangle$, $n \geq 2$,
Result: a merged set of goal $\mathcal{G}'$
create a list $SocialGoalList$, $size(SocialGoalList) = n$ ;
create a list $GoalList$, $size(GoalList) = card(\mathcal{G}_1) + ... + card(\mathcal{G}_n) - n$;
// Loop for separately grouping social goals and individual
   goals
①for $i \leftarrow 1$ to $n$ do
    for $j \leftarrow 1$ to $card(\mathcal{G}_i)$ do
        $\langle t_c, f_s\rangle \leftarrow g_j$;
        if $g_j = socialgoal$ then
            add $\langle t_c, f_s\rangle$ to $SocialGoalList$;
        else
            add $\langle t_c, f_s\rangle$ to $GoalList$
$\langle t_c, f_s\rangle \leftarrow SocialGoalList[1]$;
②for $i \leftarrow 2$ to $n$ do
    $\langle t_{ci}, f_{si}\rangle \leftarrow SocialGoalList[i]$;
    $t_c \leftarrow OR\_composition(t_c, t_{ci})$;
    $f_s \leftarrow AND\_composition(f_s, f_{si})$;
$g_{social} \leftarrow \langle t_c, f_s\rangle$;
$Temp\_GoalList \leftarrow GoalList$;
/* Loop for finding equivalent goals                              */
③ for $i \leftarrow 1$ to $size(GoalList)$ do
    $Equivalent\_GoalList \leftarrow \varnothing$;
    $\langle t_c, f_s\rangle \leftarrow GoalList[i]$;
    for $j \leftarrow 1$ to $size(Temp\_GoalList)$ do
        $\langle t_{cj}, f_{sj}\rangle \leftarrow GoalList[j]$;
        if $(GoalList[i] \neq GoalList[j]) \wedge (f_{sj} \Leftrightarrow f_s)$ then
            add $GoalList[j]$ to $Equivalent\_GoalList$;
            remove $GoalList[j]$ from $Temp\_GoalList$;
    if $size(Equivalent\_GoalList) \neq 0$ then
        for $h \leftarrow 1$ to $size(Equivalent\_GoalList)$ do
            $\langle t_{ch}, f_{sh}\rangle \leftarrow Equivalent\_GoalList[h]$;
            $t_c \leftarrow AND\_composition(t_c, t_{ch})$;
    $g \leftarrow \langle t_c, f_s\rangle$;
    add $g$ to $Final\_GoalList$;
$G' \leftarrow Final\_GoalList$
```

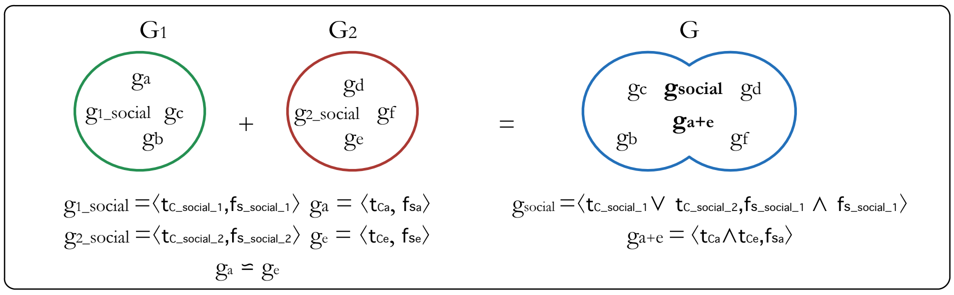

G1
G2
G
ga
g1_social gc
gb
gd
g2_social gf
ge
+
=
gc gsocial gd
ga+e
gb gf
g1_social =⟨tc_social_1,fs_social_1⟩ ga = ⟨tca, fsa⟩
g2_social =⟨tc_social_2,fs_social_2⟩ ge = ⟨tce, fse⟩
ga = ge
gsocial =⟨tc_social_1∨ tc_social_2,fs_social_1 ∧ fs_social_1⟩
ga+e = ⟨tca∧tce,fsa⟩

Figure 5: Graphical representation of Goal Fusion Algorithm (see Algorithm 2)

**Algorithm 3:** Combine norm and goal

```
Data: a set of goal $\mathcal{G}$ and several sets of norms $\mathcal{N}_i$, $i = 1...n$
Result: $\mathcal{G}_{merged}$
$GoalList \leftarrow G_{Temp}$;
create a list $NormList$, $size(NormList) = card(\mathcal{N}_1) + ... + card(\mathcal{N}_n)$;
$G_{merged} \leftarrow \varnothing$;
// Loop for creating a unique list of norms from $\mathcal{N}_1, ..., \mathcal{N}_n$
① for $i \leftarrow 1$ to $n$ do
    for $j \leftarrow 1$ to $card(\mathcal{N}_i)$ do
        $\langle r, g, \rho, \phi, d\rangle \leftarrow n_j$;
        if $n_j$ is not logically contradictory then
            add $\langle r, g, \rho, \phi, d\rangle$ to $NormList$;
        else
            revise norm
// Loop for encapsulating norms into goals
② for $i \leftarrow 1$ to $size(GoalList)$ do
    if $GoalList[i]$ is a composed goal then
        $(\phi_{mergedOR}, \phi_{mergedAND}) \leftarrow null$;
        foreach $g \subset GoalList[i]$ do
            $(\phi_{OR}, \phi_{AND}) \leftarrow compose\_norm(g, NormList)$; // See Alg.4
            $\phi_{mergedOR} \leftarrow OR\_composition(\phi_{mergedOR}, \phi_{OR})$;
            $\phi_{mergedAND} \leftarrow AND\_composition(\phi_{mergedAND}, \phi_{AND})$;
    else
        $(\phi_{mergedOR}, \phi_{mergedAND}) \leftarrow$
        $compose\_norm(GoalList[i], NormList)$;
    // Goal composition
    $\langle t_c, f_s\rangle \leftarrow GoalList[i]$;
    $t_c \leftarrow OR\_composition(t_c, \phi_{mergedOR})$;
    $t_c \leftarrow AND\_composition(t_c, \phi_{mergedAND})$;
    add $\langle t_c, f_s\rangle$ to $G_{merged}$;
```

$n = \langle \_, g, \rho, \varphi, d \rangle$ $g = \langle t_c, f_s \rangle$

A) Permission

| $t_c \vee \varphi$ | $t_c$ | $\varphi$ |
|---|---|---|
| 1 | 1 | 0 |
| 1 | 0 | 1 |
| 0 | 0 | 0 |
| 1 | 1 | 1 |

B) Obligation

| $t_c \vee (\varphi \wedge t_c)$ | $t_c$ | $\varphi$ |
|---|---|---|
| 1 | 1 | 0 |
| 0 | 0 | 1 |
| 0 | 0 | 0 |
| 1 | 1 | 1 |

C) Prohibition

| $t_c \wedge \neg\varphi$ | $t_c$ | $\varphi$ |
|---|---|---|
| 1 | 1 | 0 |
| 0 | 0 | 1 |
| 0 | 0 | 0 |
| 0 | 1 | 1 |

Figure 6: Activation Table

**Algorithm 4:** Compose Norm

```
Data: a goal $g_{current}$, a list of norms $NormList$
Result: a set $(\phi_{mergedOR}, \phi_{mergedAND})$
$List\phi\_OR \leftarrow \varnothing$;
$List\phi\_AND \leftarrow \varnothing$;
// Identification of norm types
①for $j \leftarrow 1$ to $size(NormList)$ do
    $\langle r, g, \rho, \phi, d \rangle \leftarrow NormList[j]$;
    $\langle t_c, f_s \rangle \leftarrow g$;
    // Choose among norms of current goal, what are directly
       linked to final state of goal and that are not in
       opposition to goal
    if $(g = g_{current}) \wedge (f_s = \rho) \wedge ((f_s \wedge d(\rho)) \neq \bot)$ then
        switch $d$ do
            case Obligation
                break;
            case Prohibition
                add $\neg\phi$ to $List\phi\_AND$;
            case Permission
                add $\phi$ to $List\phi\_OR$;
// Permissions give alternatives (OR)
②if $Size(List\phi\_OR) \neq 0$ then
    $\phi_{mergedOR} \leftarrow List\phi\_OR[1]$;
    for $h \leftarrow 2$ to $Size(List\phi\_OR)$ do
        $\phi_{mergedOR} \leftarrow OR\_composition(\phi_{mergedOR}, List\phi\_OR[h])$;
// Prohibition are mandatory (AND)
③if $Size(List\phi\_AND) \neq 0$ then
    $\phi_{mergedAND} \leftarrow List\phi\_AND[1]$;
    for $h \leftarrow 2$ to $Size(List\phi\_AND)$ do
        $\phi_{mergedAND} \leftarrow AND\_composition(\phi_{mergedAND}, List\phi\_AND[h])$;
```

**Algorithm 5:** Pursuable Goal

```
Data:  a goal $g$
Result: Boolean
$\langle t_c, f_s \rangle \leftarrow g$;
$List\_Of\_Atomic\_Formulae \leftarrow decompose(t_c)$;
remove multiple instances from $List\_Of\_Atomic\_Formulae$;
/* Generate all potential models of $\mathcal{W}$                    */
$List\_Of\_\mathcal{W} \leftarrow \varnothing$;
for $k \leftarrow 1$ to $List\_Of\_Atomic\_Formulae$ do
    $List\_Of\_\mathcal{W}[k] \leftarrow Combination(n, k)$;
foreach $\mathcal{W} \in List\_Of\_\mathcal{W}$ do
    if $t_c(\mathcal{W}) = true$ then
        return $true$;
return $false$;
```

## *4.1. Merging Scenarios*

This section shows some elementary merging patterns. Their composition covers a wide range of merging scenarios. For the sake of clarity, let us suppose two emergency procedures *Procedure_1* and *Procedure_2* are composed of atomic activities. They are represented by the BPMN models shown in the right side of Figure 7 along with the related goal models in the left side. In these goal models, goal relationships refer to dependencies among goals. Particularly, a dependency may be between trigger conditions and final states or between final states. An arrow from a trigger condition to a final state means that the trigger condition depends on the final state. An arrow from a final state to another means the source final state depends on the target final state. In the following, some possible merging patterns are illustrated.

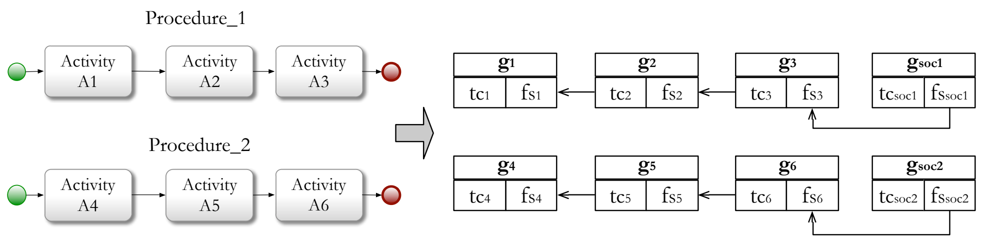


Figure 7: BPMN and goal models of dummy emergency procedures.

***CASE 1**: **Merging two procedures without common activities**.* Let us to suppose procedures *Procedure_1* and *Procedure_2* do not share common activities. This is the most simple situation that could be occur. In this case, the output of the Workflow-merge algorithm is represented by the goal model shown in the left side of Figure 8, where $t_{c_{socM}} = t_{c_1} \vee t_{c_4}$ and $f_{s_{socM}} = f_{s_3} \wedge f_{s_6}$. The right side of Figure 8 shows a possible BPMN model resulting from this goal model[6].

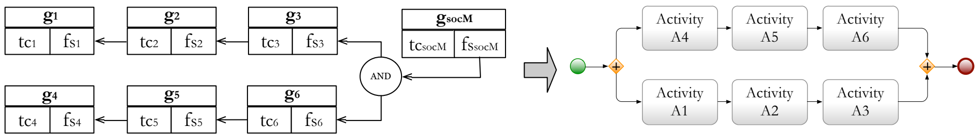


Figure 8: Goal model and BPMN model resulting from merging two procedures without common activities.

***CASE 2**: **Merging two procedures with the same initial activity**.* Let us to suppose that procedures *Procedure_1* and *Procedure_2* have the same initial activity (*Activity A1* = *Activity A4* in Figure 7). In this case, the output of the Workflow-merge algorithm is represented by the goal model shown in the right part of Figure 9, where: *(i)* the merged social goal is determined by $t_{c_{socM}} = t_{c_1} \vee t_{c_4}$ and $f_{s_{socM}} = f_{s_3} \wedge f_{s_6}$; *(ii)* the goal $g_M$ resulting from the fusion of $g_1$ and $g_4$ is determined by $t_{c_M} = t_{c_1} \wedge t_{c_4}$ and $f_{s_M} = f_{s_1}$ (or $f_{s_4}$). The bottom part of Figure 9 shows a possible BPMN model resulting from this goal model.

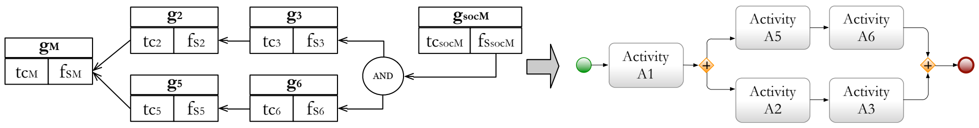


Figure 9: Goal Model and BPMN model resulting from merging two procedures with common initial activities.

***CASE 3**: Merging two procedures with common final activities.* Let us to suppose that protocols *Protocol_1* and *Protocol_2* have the same final activities, in the example shown in Fig.7 *Activity A3* equal to *Activity A6*. In this case, the output

[6]It is out of the scope of this paper the algorithm that converts a goal model in a BPMN model. It is not fundamental for this work because the WFMS that is considered in this paper executes a goal model not a BPMN model. We show the resulting BPMN model only for the sake of clarity.

of the Workflow-merge algorithm is represented by the goal model shown in the left side of Figure 10, where: *(i)* the merged social goal is determined by $t_{c_{socM}} = t_{c_3}$ and $f_{s_{socM}} = f_{s_3}$; *(ii)* the goal $g_M$ resulting from the fusion of $g_3$ and $g_6$ is determined by $t_{c_M} = t_{c_3} \wedge t_{c_6}$ and $f_{s_M} = f_{s_3}$ (or $f_{s_6}$). The right side of Figure 10 shows a possible BPMN model resulting from this goal model.

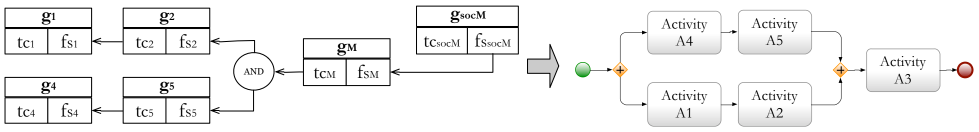


Figure 10: Goal Model and BPMN model resulting from merging two procedures with common final activities.

***CASE 4***: *Merging two procedures with initial activity equal to final activity.* Let us to suppose that the final activity of procedure *Protocol_1* is the same of the initial activity of*Protocol_2*, in the example shown in Fig.7 *Activity A3* equal to *Activity A4*. In this case, the output of the Workflow-merge algorithm is represented by the goal model shown in the left side of Figure 11, where: *(i)* the merged social goal is determined by $t_{c_{socM}} = t_{c_1} \vee t_{c_4}$ and $f_{s_{socM}} = f_{s_3}$; *(ii)* the goal $g_M$ resulting from the fusion of $g_3$ and $g_4$ is determined by $t_{c_M} = t_{c_3} \wedge t_{c_4}$ and $f_{s_M} = f_{s_3}$ (or $f_{s_4}$). The right side of Figure 11 shows a possible BPMN model resulting from this goal model.

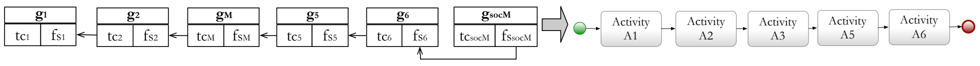


Figure 11: Goal Model and BPMN model resulting from merging two procedures with common final activities.

***CASE 5***: *Merging two different protocols with common activities.* Let us to suppose that protocols *Protocol_1* and *Protocol_2* share common activities. In particular, let us to suppose that *Activity A5* is equal to *Activity A3*. In this case, the output of the Workflow-merge algorithm is represented by the goal model shown in the top part of Figure 12, where: *(i)* the merged social goal is determined by $t_{c_{socM}} = t_{c_1} \vee t_{c_4}$ and $f_{s_{socM}} = f_{s_6}$; *(ii)* the goal $g_M$ resulting from the fusion of $g_5$ and $g_3$ is determined by $t_{c_M} = t_{c_5} \wedge t_{c_3}$ and $f_{s_M} = f_{s_5}$ (or $f_{s_3}$). The bottom part of Figure 12 shows a possible BPMN model resulting from this goal model.

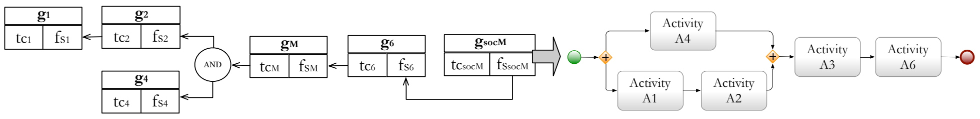


Figure 12: Goal Model and BPMN model resulting from merging two different protocols with common activities.

***CASE 6****: Merging two different protocols with common activities regulated by composition norms.* Let us to suppose that protocols *Protocol_1* and *Protocol_2* share common activities. In particular, let us to suppose that *Activity A5* is equal to *Activity A3*. Moreover, let us to suppose that for relaxing some constraints a domain expert introduces the following SBVR rule: *It is permitted that the Activity A3 is performed after Activity A4 and before Activity A2.* It is a composition norm that in our approach is represented by:

$$n = \langle \_, g_3, done(Activity\_A3), before(done(Activity\_A2)), permission\rangle$$

In such case, the output of the Workflow-merge algorithm is represented by the goal model shown in the top part of Figure 12, where: *(i)* the merged social goal is determined by $t_{c_{socM}} = t_{c_1} \vee t_{c_4}$ and $f_{s_{socM}} = f_{s_6} \wedge f_{s_2}$; *(ii)* the goal $g_M$ resulting from the fusion of $g_5$ and $g_3$ is determined by $t_{c_M} = t_{c_5} \wedge (t_{c_3} \vee before(done(Activity\_A2)))$ and $f_{s_M} = f_{s_5}$ (or $f_{s_3}$). The bottom part of Figure 13 shows a possible BPMN model resulting from this goal model.

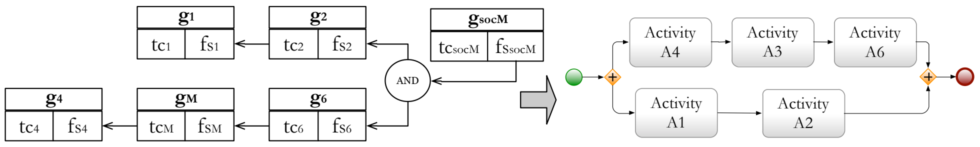


Figure 13: Goal Model and BPMN model resulting from merging two different protocols with common activities regulated by norms .

## 5. Discussions

In this paper we have defined a workflow merging algorithm based on a Goal-Norm specification of processes. In the following we discuss strengths and limits of the proposed approach.

**Detection of Merging Points -** A key step of workflow merging is to find merging points. A weakness of the approach proposed by Sun *et.al* lies on manual detection of merging points. Our approach overcomes such limitation by introducing a specification of a process in term of goals. In particular we introduced the concept of *equivalent goal* that allows to identify goals whose fulfilment produces the same effect on the state of the world. The formal specification of goals and their equivalence allowed us to implement algorithms that automatically detect merging points. We are also expand the definition of equivalent goal in such away that it will be possible to automatically find groups of activities in a workflow that could be merged with a single activity of another workflow. Moreover we are working for introducing also the concept of *semi-equivalent goal*. This concept will help to identify activities that could be partially merged by identifying goals whose fulfilment produces similar state of the world.

**Activities Conflicts Resolution -** Several merging points and norms increase the complexity of the merging process. Merging workflows have conflicting dependencies between the same pair of activities. The approach proposed in Sun *et.al* resolve manually this conflict before performing the merging process. Our approach addresses these kind of conflict by introducing opportunely defined norms, such as a *permission* in our case study. Such norms allow to automatically resolve such conflicts during the merging process. Another kind of conflict our approach is able to detect during the merging process is the presence of activity preconditions that can be never satisfied, thus making the merged workflow inconsistent. In this case our approach detect the conflict but it is not able to automatically solve it.

**Norms Merging -** The approaches proposed in literature do not address merging of business rules. Processes are commonly defined both by specifying the flow of the activities they are composed of but also rules that constraints the activities. Our merging approach considers also these restrictions by introducing norms. The concept of norm with its formal specification allow us to reason also about constraints and rules (or commonly business rules) during the merging process thus unifying them in the new merged process.

**Norm Conflicts -** Considering norms during the merging process may cause conflict about norms defined for the same activity in the merging workflows. In our approach we introduced mechanisms that allow to detect conflicts among norms. In particular, we are able to identify two kinds of conflicts that can occur: *(i)* conflicts among norms when the merging of two or more norms generates a new norm that is logically contradictory; *(ii)* conflicts between norm and goal when a norm is in opposition to a goal (i.e: the fulfilment of goal causes always a norm violation).

**Semantic Ambiguity-** Different workflows may be defined according to different set of domain concepts. Thus same concept may have the different semantic meaning in each domain as well as different concepts may indicate the same semantic meaning. This may cause semantic ambiguity when performing workflow merging. At the moment our approach is not able to cope with this issue.

## References

[1] G. Haddow, J. Bullock, D. P. Coppola, Introduction to emergency management, Butterworth-Heinemann, 2013.

[2] S. Sun, A. Kumar, J. Yen, Merging workflows: A new perspective on connecting business processes, Decision Support Systems 42 (2) (2006) 844–858.

[3] M. La Rosa, M. Dumas, R. Uba, R. Dijkman, Business process model merging: An approach to business process consolidation, ACM Transactions on Software Engineering and Methodology (TOSEM) 22 (2) (2013) 11.

[4] M. Cossentino, C. Lodato, S. Lopes, L. Sabatucci, Musa: a middleware for user-driven service adaptation.

[5] S. A. White, et al., Business process modeling notation, Specification, BPMI. org.

[6] O. M. Group, Semantics of business vocabulary and business rules (sbvr). version 1.3. may 2015.

[7] G. Witt, Writing Effective Business Rules: A Practical Method, Elsevier, 2012.

[8] L. Sabatucci, M. Cossentino, From means-end analysis to proactive means-end reasoning, in: Proceedings of the 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, IEEE Press, 2015, pp. 2–12.

[9] L. Sabatucci, P. Ribino, C. Lodato, S. Lopes, M. Cossentino, Goalspec: A goal specification language supporting adaptivity and evolution, in: Engineering Multi-Agent Systems, Springer, 2013, pp. 235–254.

[10] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, J. Mylopoulos, Tropos: An agent-oriented software development methodology, Autonomous Agents and Multi-Agent Systems 8 (3) (2004) 203–236.