Vol. 1, No. 3, pp 1 - 22, 2020
DOI: 10.55969/paradigmplus.v1n3a1

PARADIGM Plus

A Language and Methodology based on Scenarios, Grammars and Views, for Administrative Business Processes Modelling

Milliam Maxime Zekeng Ndadji [1,2,✉], Maurice Tchoupé Tchendji [1,2], Clémentin Tayou Djamegni [1], and Didier Parigot[3]

[1]*University of Dschang, Dschang, Cameroon*
{ndadji.maxime, maurice.tchoupe}@univ-dschang.org, dtayou@yahoo.com
[2]*FUCHSIA Research Associated Team, https://project.inria.fr/fuchsia/*
[3]*Inria, Sophia Antipolis, France*
didier.parigot@inria.fr

Abstract

In Business Process Management (BPM), process modelling has been solved in various ways. However, there are no commonly accepted modelling tools (languages). Some of them are criticized for their inability to capture both the lifecycle, informational and organizational models of processes. For some others, process modelling is generally done using a single graph; this does not facilitate modularity, maintenance and scalability. In addition, some of these languages are very general; hence, their application to specific domain processes (such as administrative processes) is very complex. In this paper, we present a new language and a new methodology, dedicated to administrative process modelling. This language is based on a variant of attributed grammars and is able to capture the lifecycle, informational and organizational models of such processes. Also, it proposes a simple graphical formalism allowing to model each process's execution scenario as an annotated tree (modularity). In the new language, a particular emphasis is put on modelling (using "views") the perceptions that actors have on processes and their data.

Keywords: Administrative Process Modelling · LSAWfP · Grammars · Artifact · Accreditation

Received: 2 September 2020 · Accepted: 24 October 2020 · Published: 25 October 2020.

1 Introduction

Workflow technology also known as Business Process Management (BPM) technology, aims at automating business processes. A *business process* is a set of tasks that follow a specific pattern and are executed to achieve a specific goal [1]. When such processes are managed electronically, they are called *workflows*. To automate business processes, workflow technology provides a clear framework composed of two major entities: (1) a *workflow language* for the description of such processes in a (generally graphical) format that can be interpreted by (2) a software system called *Workflow Management System* (WfMS). The role of WfMS is to facilitate collaboration and coordination of various actors involved in the distributed execution of processes' tasks: in this way, workflow technology reduces the

automation of business processes to their modelling in *workflow languages*; process modelling (specification) is therefore a crucial phase of workflow management[1].

Several tools have been developed to address process modelling. Among the most well-known are the BPMN standard [2] and the YAWL language (*Yet Another Workflow Language*) [1, 3]. Despite the significant research progress around these tools (often qualified as "*traditional tools*"), they are not unanimously accepted. Indeed, they are often criticized for not being based on solid mathematical foundations [4], for having a much too great expressiveness compared to the needs of professionals in the field (this complicates their handling and increases the related costs) [5] and/or for not being intuitive [4].

Another important criticism often levelled at traditional workflow languages is the fact that they treat data (process *information model*) and users (part of process *organizational model*) as second-class citizens by highlighting tasks and their routing (process *lifecycle model*). To precisely remedy this, researchers have developed over the last two decades and under the initiative of IBM, the artifact-centric [6] approach to the design and execution of business processes. This one proposes a new approach to workflow management by focusing on both automated processes and data manipulated using the concept of "*business artifact*" or "*artifact*" in short. A major shortcoming of artifact-centric models is that, after designing a given business process, it's difficult to manage it out of the context for which it was designed: specification and execution contexts (the WfMS on which it must be executed) are strongly coupled. In fact, in artifact-centric approaches the process specification is done with artifact modelling and artifacts are usually tailored to dedicated collaborative systems; process designers are then obliged to take into account certain details related to the workflow execution technique during the modelling phase: it is therefore difficult to consider these approaches exclusively as business process modelling tools since they are execution-context dependant.

Another mentioned shortcoming of existing process modelling approaches is that they concentrate the modelling of a given process into a single task graph. This does not allow designers to explicitly express the entire control flow of certain types of processes; in addition, the resulting specifications are generally not easy to read, to maintain and to evolve. These concerns were first raised by Wil M. P. van der Aalst et al. [7, 8]. All these shortcomings of traditional workflow languages confirm that there is still a need of scientific innovation in the field of business process modelling.

This paper presents a new *Language for the Specification of Administrative Workflow Processes* (LSAWfP) based on the concept of attributed grammars. LSAWfP is built in a more traditional way and then, unlike the artifact-centric approaches, it allows process modelling independently of a workflow execution technique. Opposed to traditional workflow languages, LSAWfP provides coherent tools to model both processes' lifecycle model, information model and organizational model. LSAWfP is particularly interested in administrative process[2] modelling as this type of process is the most frequently encountered in organizations [9, 10]. A given administrative process is naturally composed of a set of execution scenarios; an *execution scenario* (or simply *scenario*) is an ordered subset of activities that, once executed, lead the process to one of its end states, whether or not a business goal is achieved. In LSAWfP, the execution scenarios of a given administrative process are represented by a finite set $\{\mathcal{S}_{ad}^1, \ldots, \mathcal{S}_{ad}^k\}$ of so-called *representative scenarios* known in advance; representative scenario refers to any execution scenario that, in "combination" with some other representative scenarios, can generate a (potentially infinite) set of other scenarios (see sec. 3.2.1). Therefore, LSAWfP uses the *scenario* as the modelling unit: a given process modelling consists to the modelling of each of its execution scenarios. Designers can thus focus on the modelling and the maintenance of process' parts rather than handling the whole process at a time: this seems to be more intuitive, modular and easier.

To use LSAWfP, the process to be modelled must be well understood by the designer; its tasks

[1] The *Workflow Management Coalition* (it is the organization responsible for developing standards in workflow) defines *workflow management* as the modelling and computer management of all the tasks and different actors involved in executing a business process [1].

[2] These are processes for which the set of tasks (executed by humans or not) as well as their order of execution are known in advance [9, 10].

and their sequences, the data they produce and the actors taking part in their execution must be known in advance (administrative processes). In addition to these elements related to the lifecycle, the information model and the organizational model of the processes, the designer must be able to identify its various execution scenarios. The modelling approach (methodology) of LSAWfP can be described as follows: from the observation that one can analyse the textual description of a given administrative process to exhibit all its possible representative scenarios leading to its business goals, LSAWfP proposes to model each of these scenarios by an annotated tree called a *representative artifact* in which, each node corresponds to a task of the process, and each hierarchical decomposition (a node and its sons) represents a scheduling of these tasks. From these representative artifacts, are derived an attributed grammar $\mathbb{G}$ called the *Grammatical Model of Workflow* (GMWf). The symbols of a given GMWf represent the process tasks and each of its productions represents a scheduling of a subset of these tasks; intuitively, a production given by its left and right hand sides, specifies how the task on the left hand side precedes (must be executed before) those on the right hand side. Thus, the GMWf of a process contains both its *information model* (modelled by its attributes) and its *lifecycle model* (thanks to the set of its productions). Once the GMWf is obtained, LSAWfP propose to add organizational information (*organizational model*) modelled by two lists: $\mathcal{L}_{P_k}$ which contains actors involved in the process and $\mathcal{L}_{\mathcal{A}_k}$ which contains their *accreditations*. These lists aim at modelling actors, their roles and the different perceptions they have on a given process. Thus, with LSAWfP, the model (subsequently called *a Grammatical Model of Administrative Workflow Process* - GMAWfP -) of a given administrative process $\mathcal{P}_{ad}$ is an executable grammatical specification given by a triplet $\mathbb{W}_f = (\mathbb{G}, \mathcal{L}_{P_k}, \mathcal{L}_{\mathcal{A}_k})$.

The rest of this manuscript is organised as follows: after presenting some basic concepts, some related works and a running example (the peer-review process) in section 2, we present more formally and with illustrations, the proposed language in section 3 and we discuss its expressiveness. A presentation of some ongoing works is conducted in section 4; in particular, we briefly present one of our current works that reinforces the justification of the need to produce a new workflow language. Finally, section 5 is devoted to the conclusion.

2 Preliminaries and Related Works

In this section, we present some basic concepts related to workflow technology to facilitate the understanding of this paper. We then give a very brief state of the art on process modelling techniques. We finally introduce a process that will be used for illustration purposes throughout this paper.

2.1 Some Basic Concepts

Workflow typology: in the literature, there are several approaches to workflow classification. However, it is the approach that classifies them by the nature and the behaviour of automated processes that is most commonly used. According to the latter, workflows are divided into three groups: production workflows, administrative workflows and ad-hoc workflows [9, 10]. Production workflows are those automating highly structured processes that experience very little (or no) change over time. Administrative workflows apply to processes of which all cases are known; that means that tasks are predictable and their sequencing are simple and clearly defined. Ad-hoc workflows are more general; they automate occasional processes for which it is not always possible to define all the rules in advance.

The language presented in this paper is especially tailored for administrative workflows. One of the inherent characteristics of administrative business processes is the confidentiality that must sometimes be guaranteed on data and/or tasks that are executed. It is indeed easy to imagine administrative processes in which, various actors at any given time, have only a potentially partial perception of all the activities that have already and/or must be carried out: the perception that an actor has on the current state of a process is called his "view on the process". For example, in a peer-review process, a

reviewer does not necessarily need to know if another reviewer has been contacted for the expertise of the article entrusted to him; and even if so, he should not necessarily know if the latter has already returned his report, etc. Similarly, when organising a journey for a Head of State, not all actors (secret services, civil office, doctor, presidential guard, etc.) have access to the same information which may include for example, tasks to be executed, their dates and states of execution, etc. Administrative workflows are characterized by the fact that all cases (tasks and their sequences), all actors and the permissions they have on tasks, etc. are known in advance. When specifying such processes, it should also be possible to model confidentiality constraints; for example, it should be possible to explicitly express the permissions which each actor has on each task. In LSAWfP, this requirement is treated as first-order concern with the help of a model called "accreditation", which allows to materialize the perception of each actor on the processes and their data.

Business process specification: the specification of a business process is commonly referred to as a *workflow model*. According to [11], a workflow model consists of three main conceptual models: the *organizational*, *informational* and *lifecycle* models. The *organizational model* is used to express and classify the resources responsible for executing the tasks of the studied process. Generally, these are classified into *roles* to which tasks are assigned. The *informational model* is used to describe the structure of consumed and produced data during processes execution. Finally, the *lifecycle model* is used to describe the structure of each task, the coordination between them and consequently, the coordination between the various actors involved in their execution. The lifecycle model is generally expressed using a language and allows the expression of basic control flows (*sequential*, *parallel*, *alternative* and *iterative*) between tasks. Ideally, a workflow language should be able to allow workflow model designers to express these three conceptual models.

2.2 Related Works

There is a lot of work that has been already done in terms of process modelling. So, there is a plethora of workflow languages; the few presented here are intended to interest the reader.

In the survey [12], the authors group workflow languages into two categories: those based on graphical models (graph-based formalism), and the others based on rule specifications (rule-based formalism). In graph-based workflow languages, processes are specified using graphical models where tasks are represented as nodes, and control flow and data dependencies between tasks as arcs. In rule-based workflow languages, process logic is coded as a set of rules, each of which is associated with one or more business tasks and specifies their properties such as their pre and post conditions of execution.

BPMN and YAWL are graph-based workflow languages. The BPMN standard [2] was initiated by the *Business Process Management Initiative* (BPMI) which merged with *Object Management Group* (OMG) in 2005. It is a simple formalism inspired by the statecharts. BPMN is informal: i.e, it does not have well-defined semantic, so, resulting specifications are difficult to analyse [4]. The YAWL language (*Yet Another Workflow Language*) [1, 3] is based on a formalism called WF-Net (Workflow Net) [10], derived from that of Petri nets. Unlike BPMN, YAWL has a solid mathematical basis that facilitates the automatic analysis of its process models.

As rule-based workflow languages, we can mention Event-Condition-Action (E-C-A) Business Rules [13] and ADEPT [14]. The language in [13] is based on the E-C-A paradigm[3]; an E-C-A rule-based process model to serve as an integration layer between multiple process modelling languages is provided. In the ADEPT multi-agent system, process logic is expressed in the so-called service definition language (SDL); the resulting model is such as at runtime, agents have sufficient freedom to take alternative execution paths (from the model) to complete the process goal.

[3] E-C-A is a paradigm that specifies the desired behaviour for reactive systems (i.e. systems that maintain ongoing interactions with their environments). In such a system centered around the E-C-A paradigm, when an event occurs, a condition is evaluated (by a querying mechanism) and the system takes corresponding action [15].

As mentioned in the introduction, for the last two decades, a lot of work on process modelling has been done on the artifact-centric paradigm [6, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26]. This paradigm was introduced by IBM through the work of Nigam and Caswell [6]. It recommends that, when modelling processes, one should focus on modelling a data structure called artifact that can give information both on the execution state of a process instance at a given time, and on the "how" to make this state evolve. Hull et al. [18] extends the artifact-centric model of [6] to provide an interoperation framework in which data are hosted on central infrastructures named *artifact-centric hubs*. They propose mechanisms (including user views) for controlling access to these data. Lohmann and Wolf [19] provide a choreography-like framework for artifact-centric interoperation. They abandon the fact of having a single artifact hub [18] and they introduce the idea of having several agents which operates on artifacts. Some of those artifacts are mobile; thus, the authors provide a systematic approach for modelling artifact location and its impact on the accessibility of actions using a Petri net. Badouel et al. [23, 24] introduce a flexible framework for data-centric case management. Their model puts stress on modelling process data and users as first class citizens. As for LSAWfP, they use an attributed grammar (named Guarded Attribute Grammar - GAG -) as the mathematical foundation of their model.

Some of the foundations of the artifact-centric paradigm come from the proclet model [7, 8]. In the latter, the authors provide a solution to the uniqueness of the task graph (which makes it unreadable and difficult to maintain) when modelling a given process. They introduce the concept of *proclet*; they thus propose to deal with several levels of granularity assigned to lightweight workflow processes (proclets) in charge of orchestrating their execution. The modelling of each level of granularity is therefore done using a smaller task graph. We find this vision very interesting. However, the notion of granularity manipulated in [7] is not very intuitive and seems, as for artifact-centric models, intimately linked to the execution model of proclets. In the case of an administrative process $\mathcal{P}_{ad}$, we think it would be more affordable to partition its task graph according to a characteristic that is natural to it like its execution scenarios. Knowing that such a process is naturally composed of a set of execution scenarios and can be represented by a finite set $\{\mathcal{S}_{ad}^1, \ldots, \mathcal{S}_{ad}^k\}$ of representative scenarios (see sec. 3.2.1) known in advance, we propose to use the scenario (execution scenario) as the modelling unit.

With the advent of cloud computing, several studies on business process execution have proposed completely decentralized models that can be deployed on Peer-to-Peer architectures [27, 28, 29]. The most current solutions tend to have the processes executed by blockchain-based systems [30, 31, 32, 33, 34, 35, 36]. There is therefore a need to adapt workflow languages so that the processes they specify can be executed in a distributed manner on such systems. The language we propose is in line with this need.

2.3 A Running Example: the Peer-Review Process

As running example, we will use the peer-review process. A brief description of it inspired by those made in [7, 23, 25, 26], can be the following one:

- The process starts when the editor in chief (EC) receives a paper for validation;
- Then, the EC performs a pre-validation after which he can accept or reject the submission for various reasons (subject of minor interest, submission not within the journal scope, non-compliant format, etc.); let us call this **task** "A";
- If he rejects the submission, he writes a report (**task** "B") then notifies the corresponding author (**task** "D") and the process ends;
- Otherwise, he chooses an associated editor (AE) and sends him the paper for the continuation of its validation;

– The AE prepares the manuscript (**task** "C") and contacts simultaneously two experts for the evaluation of the paper (**tasks** "$E1$" and "$E2$"); if a contacted expert refuses to participate, the AE contacts another one (iteration on **task** "$E1$" or "$E2$"). Otherwise, the expert (referee) can start the evaluation;

– Each referee reads, seriously evaluates the paper (**tasks** "$G1$" and "$G2$") and sends back a report (**tasks** "$H1$" and "$H2$") and a message (**tasks** "$I1$" and "$I2$") to the AE;

– After receiving reports from all referees, the AE takes a decision and informs the EC (**task** "F") who sends the final decision to the corresponding author (**task** "D").

From the description above, one can identify all the tasks to be executed, their sequencing, actors involved and the tasks assigned to them. For this case, four actors are involved: an editor in chief (EC) who is responsible for initiating the process, an associated editor (AE) and two referees ($R1$ and $R2$). Figure 1 shows the orchestration diagrams corresponding to the graphical description of this peer-review process using BPMN (*Business Process Model and Notation*) and WF-Net (*Workflow Net*). Each diagram resumes the *main scenarios* of the studied process. The purpose of this figure is to show the independence of our running example from a specific workflow language; in addition, we want the reader to bear in mind what the two workflow languages used in this figure offer in terms of process modelling tools and methodology in order to better understand the contributions of LSAWfP. It can be seen that with both formalisms, the modelling of all scenarios is tackled at the same time and process data is treated in background or not at all. With BPMN, the organizational aspect highlights the actors and the tasks assigned to them, but nothing more. The organizational aspect is absent with the WF-Net formalism; however, the control flow is more expressive because it highlights the different states of the process after the execution of each task. We have added a small caption for the uninformed reader; the bibliographical references on the used languages can also be a valuable aid to the understanding of this figure.

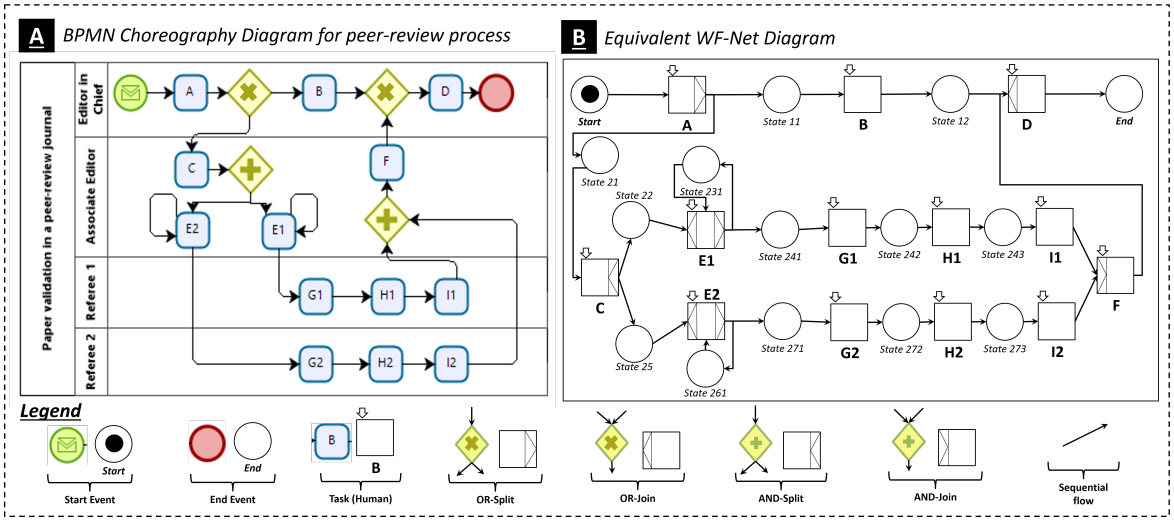

Figure 1: Orchestration diagrams of the peer-review process.

3 A Language for the Specification of Administrative Workflow Processes (LSAWfP)

In this section, we present the new language LSAWfP that allows to specify administrative workflow processes independently of a workflow execution technique, and with the use of scenario as modelling unit.

3.1 Artifacts as Control Flow Graphs

Let's consider an administrative process $\mathcal{P}_{ad}$ to be modelled. By definition (of administrative process), its set $\mathbb{T}_n = \{X_1, \ldots, X_n\}$ of tasks is known in advance. In traditional workflow languages like BPMN or WF-Net, the control flow between its tasks is represented using a directed graph that can contain cycles (see Figure 1). Such a graph allows the modelling of the potentially infinite set[4] of $\mathcal{P}_{ad}$'s execution scenarios. Let's note however that each $\mathcal{P}_{ad}$'s execution scenario can also be modelled using an annotated tree t_i called *artifact*. Indeed, starting from the fact that a given scenario $\mathcal{S}^i_{ad}$ consists of a subset $\mathbb{T}_m \subseteq \mathbb{T}_n$ of $m \leq n$ tasks whose instances are to be executed in a specific order (in parallel or in sequence), one can represent $\mathcal{S}^i_{ad}$ as a tree t_i in which each node (a task instance labelled X_i) potentially corresponds to a task $X_i \in \mathbb{T}_m$ of $\mathcal{S}^i_{ad}$ and each hierarchical decomposition (a node and its sons) corresponds to a scheduling: the task associated with the parent node must be executed before those associated with the son nodes; the latter must be executed according to an order - parallel or sequential - that can be specified by particular annotations "$\wr$" (is sequential to) and "$\|$" (is parallel to) which will be applied to each hierarchical decomposition. The annotation "$\wr$" (resp. "$\|$") reflects the fact that the tasks associated with the son nodes of the decomposition must (resp. can) be executed in sequence (resp. in parallel). To model iteration, nodes can be recursive in an artifact: i.e a node labelled X_i may appear in sub-trees rooted by a node having the same label X_i.

Considering the running example (the peer-review process), two of its execution scenarios can be modelled using the two artifacts art_1 and art_2 in figure 2. In particular, we can see that art_1 shows how the task "Receipt and pre-validation of a submitted paper" assigned to the EC, and associated with the symbol A (see sec. 2.3), must be executed before tasks associated with the symbols B and D that are to be executed in sequence from the left to the right. Note that additional symbols called *(re)-structuring symbols* can be added in artifacts to correct the scheduling of tasks (we better explain this in section 3.2.2): this is the case for art_2 in which the symbol $S1$ has been added.

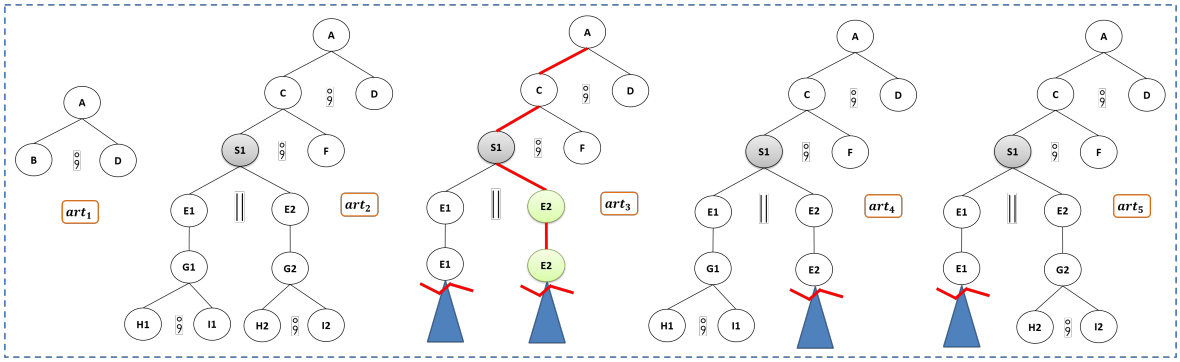

Figure 2: Representative artifacts of a paper validation process in a peer-review journal.

[4] This is the case when there is one or more iterative routing (materialized by cycles in the task graph) on tasks.

3.2 Representative Artifacts and Grammatical Model of Workflow

3.2.1 Representative Artifacts

As mentioned earlier (see sec. 3.1), the set of execution scenarios for a given administrative process can be infinite. This is the case of our running example process in which we can iterate on tasks $E1$ and $E2$ without limit; as each iteration on either $E1$ or $E2$ give rise to a new execution scenario, we thus generate an infinite set of execution scenarios. In these cases, the designer cannot list this set of scenarios in order to model each of them. This problem can be avoided by taking inspiration from the role played by the concept of vector spaces' basis in mathematics (algebra). In this sense, we can find a finite set $\tau = \{\mathcal{S}_{ad}^1, \ldots, \mathcal{S}_{ad}^k\}$ of scenarios said to be *representative* and modelled be a finite set $\zeta = \{t_1, \ldots, t_k\}$ of *representative artifacts* such that, any artifact representing an execution scenario can be expressed as a "combination" of some elements of ζ.

We designate by the expression *nominal scenario* of a given process, any scenario leading to a given business goal without iteration; in the same vein, scenarios in which at least one iteration have been made are called *alternative scenarios*. For a given process, the set of nominal scenarios is finite and the artifacts depicting each of these scenarios are part of the process representative artifacts. The other part of the process representative artifacts is determined much more technically, from the (potentially infinite) set of alternative scenarios. Concretely, when designing an alternative scenario artifact, the designer must prune it at each first iteration encountered: i.e, the designer must prune each branch of an alternative scenario artifact as soon as he encounters a node labelled for the second time, by a same label along a path starting from the root. By doing so, the designer does not redevelop a node into a subtree that has already been explored: thus, he does not loop endlessly on potential iterations and does not explore all alternative scenarios generated by those iterations. However, the resulting pruned artifact contains patterns (productions) that indicate how to construct the artifacts associated with the considered alternative scenarios: this is why the obtained pruned artifact is said to be representative.

More precisely, one could assume that to design the representative artifacts of a given business process, the designer begins by identifying the initial tasks of it (i.e., the tasks that can start one of its execution scenarios); each of these tasks will thus constitute the root of several representative artifacts. To construct the set $arts_{X_{0_i}}$ of representative artifacts rooted in a given initial task X_{0_i}, the designer will:

(1) Construct an artifact art having X_{0_i} as the single node (root);

(2) Then, he will determine the set $follow = \left\{\left(X_{1_{i_1}}, \ldots, X_{m1_{i_1}}\right), \ldots, \left(X_{1_{i_n}}, \ldots, X_{mn_{i_n}}\right)\right\}$ of task combinations (each combination is either sequential or parallel[5]) that can be immediately executed after the execution of X_{0_i}. For each combination $\left(X_{1_{i_j}}, \ldots, X_{mj_{i_j}}\right)$, the designer will replace the artifact art by a new artifact art_j obtained by expanding the node X_{0_i} of art such that in art_j, the tasks $X_{1_{i_j}}, \ldots, X_{mj_{i_j}}$ are the child nodes of X_{0_i}.

(3) It will then only remain to recursively develop (using the principle of (2)) each leaf node of the new artifacts until representative artifacts (those that describe an execution scenario in its entirety) are obtained.

This construction principle emphasizes the fact that one does not loose information by pruning an artifact when encountering a given node X for the second time in the same branch. In such a case, it is not necessary to develop X a second time since the designer has enumerated (in several artifacts) all the possibilities (scenarios) of continuing the execution of the process after the execution of the task associated with X. As we will see in section 3.2.2, these possibilities will be coded in a grammar and

[5] If a given combination $\left(X_{1_{i_j}}, \ldots, X_{mj_{i_j}}\right)$ is sequential (resp. parallel), its tasks are to be (resp. can be) executed sequentially (resp. in parallel).

thus, the execution scenarios characterized by several iterations on X, will indeed be specified in the language. When constructing a representative artifact, the pruning of a branch is therefore systematic when a node is encountered for the second time; no matter how many nodes generate an iteration in the same branch.

Figure 2 presents the five representative artifacts of our running example process. The artifacts art_1 and art_2 model the two nominal scenarios: art_1 models the scenario in which the EC directly rejects the paper while art_2 models the case where the paper is evaluated by referees ($R1$ and $R2$) without the AE having to contact more than two experts (no iteration on tasks $E1$ and $E2$). The artifacts art_3, art_4 and art_5 represent the infinite set of alternative scenarios in this example: some of their subtrees (those represented by blue triangles) have been pruned. For illustration purposes, we put forward the pruning made by the designer on the node $E2$ (in green colour) of art_3, which appeared for the second time in the same branch (the branch is highlighted in red colour).

3.2.2 Grammatical Model of Workflow

From the finite set of representative artifacts of a given process, it is possible to extract an abstract grammar[6] that represents the underlying process's lifecycle model : it is this grammar that we designate by the expression *Grammatical Model of Workflow* (*GMWf*).

Let's consider the set $\{t_1, \ldots, t_k\}$ of representative artifacts modelling the k representative scenarios of a given process $\mathcal{P}_{ad}$ of n tasks ($\mathbb{T}_n = \{X_1, \ldots, X_n\}$). Each t_i is a derivation tree for an abstract grammar (a GMWf) $\mathbb{G} = (\mathcal{S}, \mathcal{P}, \mathcal{A})$ whose set of symbols is $\mathcal{S} = \mathbb{T}_n$ (all process tasks) and each production $p \in \mathcal{P}$ reflects a hierarchical decomposition contained in at least one of the representative artifacts. Each production is therefore exclusively of one of the following two forms: $p : X_0 \rightarrow X_1 \fatsemi \ldots \fatsemi X_n$ or $p : X_0 \rightarrow X_1 \parallel \ldots \parallel X_n$. The first form $p : X_0 \rightarrow X_1 \fatsemi \ldots \fatsemi X_n$ (resp. the second form $p : X_0 \rightarrow X_1 \parallel \ldots \parallel X_n$) means that task X_0 must be executed before tasks $\{X_1, \ldots, X_n\}$ that must be (resp. can be) executed in sequence (resp. in parallel) from the left to the right. A GMWf can therefore be formally defined as follows:

Definition 1 *A **Grammatical Model of Workflow** (GMWf) is defined by $\mathbb{G} = (\mathcal{S}, \mathcal{P}, \mathcal{A})$ where :*

- *$\mathcal{S}$ is a finite set of **grammatical symbols** or **sorts** corresponding to various **tasks** to be executed in the studied business process;*
- *$\mathcal{A} \subseteq \mathcal{S}$ is a finite set of particular symbols called **axioms**, representing tasks that can start an execution scenario (roots of representative artifacts), and*
- *$\mathcal{P} \subseteq \mathcal{S} \times \mathcal{S}^*$ is a finite set of **productions** decorated by the annotations "$\fatsemi$" (is sequential to) and "$\parallel$" (is parallel to): they are **precedence rules**. A production $P = \left(X_{P(0)}, X_{P(1)}, \cdots, X_{P(|P|)}\right)$ is either of the form $P : X_0 \rightarrow X_1 \fatsemi \ldots \fatsemi X_{|P|}$, or of the form $P : X_0 \rightarrow X_1 \parallel \ldots \parallel X_{|P|}$ and $|P|$ designates the length of P's right-hand side. A production with the symbol X as left-hand side is called a X-production.*

Let's illustrate the notion of GMWf by considering the one generated from an interpretation of the representative artifacts for the peer-review process (see Figure 2): the derived GMWf is $\mathbb{G} = (\mathcal{S}, \mathcal{P}, \mathcal{A})$ in which the set $\mathcal{S}$ of grammatical symbols is $\mathcal{S} = \{A, B, C, D, S1, E1, E2, F, G1, G2, H1, H2, I1, I2\}$ (see sec 2.3); the only initial task (axiom) is A (then $\mathcal{A} = \{A\}$) and the set $\mathcal{P}$ of productions is[7]:

$P_1 : A \rightarrow B \fatsemi D$	$P_2 : A \rightarrow C \fatsemi D$	$P_3 : C \rightarrow S1 \fatsemi F$	$P_4 : S1 \rightarrow E1 \parallel E2$
$P_5 : E1 \rightarrow G1$	$P_6 : E2 \rightarrow G2$	$P_7 : E1 \rightarrow E1$	$P_8 : E2 \rightarrow E2$
$P_9 : G1 \rightarrow H1 \fatsemi I1$	$P_{10} : G2 \rightarrow H2 \fatsemi I2$	$P_{11} : B \rightarrow \varepsilon$	$P_{12} : D \rightarrow \varepsilon$
$P_{13} : F \rightarrow \varepsilon$	$P_{14} : H1 \rightarrow \varepsilon$	$P_{15} : I1 \rightarrow \varepsilon$	$P_{16} : H2 \rightarrow \varepsilon$
$P_{17} : I2 \rightarrow \varepsilon$			

[6] It is enough to consider the set of representative artifacts as a "set of generators" of a regular tree language: there is therefore an (abstract) grammar to generate them.

[7] A production of the form $X \rightarrow \varepsilon$ indicates that task X is not "decomposable" in subtasks.

There may be special cases where it is not possible to schedule the tasks of a scenario using the two (only) forms of production selected for GMWf. For example, this is the case for the peer-review process wherein task C precedes tasks $E1$, $E2$ and F, tasks $E1$ and $E2$ can be executed in parallel and precede F (see sec. 2.3). In such cases, the introduction of a few new symbols known as (*re*)*structuring symbols* (not associated with tasks) can make it possible to produce a correct scheduling. For the peer-review process example, the introduction of a new symbol $S1$ allows us to obtain the following productions: $P_3 : C \rightarrow S1 \fatsemi F$ and $P_4 : S1 \rightarrow E1 \parallel E2$ which properly model the required scheduling and avoid the usage of the malformed production $p : C \rightarrow E1 \parallel E2 \fatsemi F$ (see in Figure 2, art_2, the node $S1$ — in gray —). To deal with such cases, the previously given GMWf definition (definition 1) is slightly adapted by integrating the (re)structuring symbols; the resulting definition is as follows:

Definition 2 *A **Grammatical Model of Workflow** (GMWf) is defined by $\mathbb{G} = (\mathcal{S}, \mathcal{P}, \mathcal{A})$ wherein $\mathcal{P}$ and $\mathcal{A}$ refer to the same purpose as in definition 1, $\mathcal{S} = \mathcal{T} \cup \mathcal{T}_{Struc}$ is a finite set of **grammatical symbols** or **sorts** in which, those of $\mathcal{T}$ correspond to **tasks** of the studied business process, while those of $\mathcal{T}_{Struc}$ are (re)structuring symbols.*

3.3 Modelling the Information and Organization Model of Processes with LSAWfP

3.3.1 An Information Model for LSAWfP

As formalized in definition 2, a GMWf perfectly models the tasks and control flow of administrative processes (lifecycle model). In this section we discuss the specification of processes-related data (*the information model*) in LSAWfP.

It is not easy to model the structure of business processes data using a general type as they differ from one process to another. For the current work, tackling the processes data structure has no proven interest because it does not bring any added value to the proposed model since, we are not specifically interested in data modelling but rather in process modelling: a representation of these data using a set of variables associated with tasks is largely sufficient. However, it should be noted that in existing data-driven modelling approaches like the Guarded Attribute Grammar (GAG) model [24, 37, 38], each task comes equipped with a set of *inherited* attributes (terms over a ranked alphabet) and a set of *synthesised* attributes where: inherited attributes represents input data (i.e, necessary data for the associated task to be executed) while synthesised attributes represents output data (i.e, data that are produced after the task being executed). In addition, dependency relationships between data (attributes) are often specified.

In this work, the potentially manipulated data by a given process task is represented using a single *attribute* embedded in the nodes associated with it. This is more than enough to show that LSAWfP cares about processes' data (this is one of this paper's goals). In more specific (future) work, the form of this attribute can be simply refined (as in the GAG approach [24, 37, 38]) to allow designers to better describe the nature of the manipulated data and their impact on processes. To formalize the taking into account of attributes, we update for the last time the definition of GMWf. We thus associate with each symbol, an attribute named $status$ allowing to store all the data of the associated task; its precise type is left to the discretion of the process designer. The new definition of GMWf is thus the following one:

Definition 3 *A **Grammatical Model of Workflow** (GMWf) is defined by $\mathbb{G} = (\mathcal{S}, \mathcal{P}, \mathcal{A})$ wherein $\mathcal{S}$, $\mathcal{P}$ and $\mathcal{A}$ refer to the same purpose as in definition 2. Each grammatical symbol $X \in \mathcal{S}$ is associated with an attribute named **status**, that can be updated when tasks are executed; **X.status** provides access (read and write) to its content.*

3.3.2 An Organizational Model for LSAWfP

Because business processes are generally carried out collectively, it is important to model actors and to set up mechanisms to ensure better coordination between them and to eventually guarantee the confidentiality of certain actions and data: this is the purpose of *accreditation*. The accreditation of a given actor provides information on its rights (permissions) relatively to each sort (task) of the studied process's GMWf. We propose here, a simple but non-exhaustive nomenclature of rights. It is inspired by the one used in UNIX-like operating systems. Three types of accreditation are therefore defined: accreditation in reading (r), writing (w) and execution (x).

1. *The accreditation in reading* (r): an actor accredited in reading on sort X must be informed of the execution of the associated task; he must also have free access to its execution state (data generated during its execution). We call an actor's ***view***, the set of sorts on which he is accredited in reading.

2. *The accreditation in writing* (w): an actor accredited in writing on sort X can execute the associated task. The designation of the right to execute a task by the term accreditation in writing can be confusing. However, we consider that the execution of tasks is performed (manually and/or automatically) by human actors. At the end of a given task execution, the actor in charge of its execution must enter (write) the produced data into the system: he must have an accreditation in writing. To make it simple, any actor accredited in writing on a sort must necessarily be accredited in reading on it[8].

3. *The accreditation in execution* (x): an actor accredited in execution on sort X is allowed to ask the actor who is accredited in writing in it, to execute it (realization of the associated task). This right is particularly appropriate for the modelling of interaction between actors: especially in the case of processes where it is important to know "who" can ask "who" to perform a given task. This is not a "delegation" of work, since there is no transfer of tasks, or of competencies. Work delegation can be the subject of more elaborate work on the LSAWfP organizational model.

More formally, an accreditation is defined as follows:

Definition 4 *An **accreditation** $\mathcal{A}_{A_i}$ defined on the set $\mathcal{S}$ of grammatical symbols for an actor A_i, is a triplet $\mathcal{A}_{A_i} = \left(\mathcal{A}_{A_i(r)}, \mathcal{A}_{A_i(w)}, \mathcal{A}_{A_i(x)}\right)$ such that, $\mathcal{A}_{A_i(r)} \subseteq \mathcal{S}$ also called **view** of actor A_i, is the set of symbols on which A_i is accredited in reading, $\mathcal{A}_{A_i(w)} \subseteq \mathcal{A}_{A_i(r)}$ is the set of symbols on which A_i is accredited in writing and $\mathcal{A}_{A_i(x)} \subseteq \mathcal{S}$ is the set of symbols on which A_i is accredited in execution.*

The accreditations of various actors must be produced by the workflow designer just after modelling the scenarios in the form of representative artifacts. From the task assignment for the peer-review process in the running example (see sec. 2.3), it follows that the accreditation in writing of the EC is $\mathcal{A}_{EC(w)} = \{A, B, D\}$, that of the AE is $\mathcal{A}_{AE(w)} = \{C, S1, E1, E2, F\}$ and that of the first (resp. the second) referee is $\mathcal{A}_{R_1(w)} = \{G1, H1, I1\}$ (resp. $\mathcal{A}_{R_2(w)} = \{G2, H2, I2\}$). Since the EC can only execute the task D if the task C is already executed (see Figure 2), in order for the EC to be able to ask the AE to execute this task, he must be accredited in execution on it; so we have $\mathcal{A}_{EC(x)} = \{C\}$. Moreover, in order to be able to access all the information on the peer-review evaluation of a paper (task C) and to summarize the right decision to send to the author, the EC must be able to consult the reports (tasks $I1$ and $I2$) and the messages (tasks $H1$ and $H2$) of the different referees, as well as the final decision taken by the AE (task F). These tasks, added to $\mathcal{A}_{EC(w)}$[9] constitute the set $\mathcal{A}_{EC(r)} = \mathcal{V}_{EC} = \{A, B, C, D, H1, H2, I1, I2, F\}$ of tasks on which he is accredited in reading. By doing so for each of other actors, we deduce the accreditations represented in Table 1.

[8] This hypothesis, which does not reduce the expressiveness of the language, is taken only because it is estimated that actors will operate through What You See Is What You Get (WYSIWYG) tools.

[9] Recall that we consider that one can only execute what he sees.

Table 1: Accreditations of the different actors taking part in the peer-review process.

Actor	Accreditation
EC	$\mathcal{A}_{EC} = (\{A, B, C, D, H1, H2, I1, I2, F\}, \{A, B, D\}, \{C\})$
AE	$\mathcal{A}_{AE} = (\{A, C, S1, E1, E2, F, H1, H2, I1, I2\}, \{C, S1, E1, E2, F\}, \{G1, G2\})$
$R1$	$\mathcal{A}_{R1} = (\{C, G1, H1, I1\}, \{G1, H1, I1\}, \varnothing)$
$R2$	$\mathcal{A}_{R2} = (\{C, G2, H2, I2\}, \{G2, H2, I2\}, \varnothing)$

Since the (re)structuring symbols are not associated with tasks and were only introduced to adjust the control flow, their execution neither requires nor produces data; they play the same role as gateways in traditional workflow languages. Therefore, the accreditation in writing and execution on them may be best left to the designer's appreciation; he will then make the assignment by referring to the execution model he will use later. To this end, he could use the same principle for the assignment of these accreditations in the case of concrete process' tasks. However, one could by default consider that all actors are accredited in reading on (re)structuring symbols; this would make these symbols visible to all of them and would guarantee that the adjustment of the control flow will be effective for all of them even if they have partial perceptions of the process.

3.4 Summary

3.4.1 Definition of LSAWfP

To summarise, we state that in LSAWfP, an administrative process $\mathcal{P}_{ad}$ is specified using a triplet $\mathbb{W}_f = (\mathbb{G}, \mathcal{L}_{P_k}, \mathcal{L}_{\mathcal{A}_k})$ called *a Grammatical Model of Administrative Workflow Process* (GMAWfP) and composed of: a GMWf, a list of actors and a list of their accreditations. The GMWf is used to describe all the tasks of the studied process and their scheduling, while the list of accreditations provides information on the role played by each actor involved in the process execution. A GMAWfP can then be formally defined as follows:

Definition 5 *A **Grammatical Model of Administrative Workflow Process** (GMAWfP) $\mathbb{W}_f$ for a given business process, is a triplet $\mathbb{W}_f = (\mathbb{G}, \mathcal{L}_{P_k}, \mathcal{L}_{\mathcal{A}_k})$ wherein $\mathbb{G}$ is the studied process (global) GMWf, $\mathcal{L}_{P_k}$ is the set of k actors taking part in its execution and $\mathcal{L}_{\mathcal{A}_k}$ represents the set of these actors accreditations.*

3.4.2 How to Model a Process with LSAWfP (Methodology)

To model a given process using LSAWfP, one must start from a textual description of the process and perform the four activities illustrated in Figure 3. First, the set of tasks and their execution precedence relationships must be identified in order to produce a finite set of representative artifacts following the approach presented in section 3.2.1. The GMWf must then be deduced from the set of representative artifacts thus produced. Then, the different actors involved in the execution of the process being modelled must be identified, and finally, a coherent assignment of tasks to actors must be made and their respective accreditations deduced.

3.5 On the Expressiveness of LSAWfP

Let's consider a specification $\mathbb{W}_f = (\mathbb{G}, \mathcal{L}_{P_k}, \mathcal{L}_{\mathcal{A}_k})$ of a given business process $\mathcal{P}_{ad}$. As described above, its organizational model that expresses and classifies/assigns the resources that must execute its tasks is given by the couple $(\mathcal{L}_{P_k}, \mathcal{L}_{\mathcal{A}_k})$ of $\mathbb{W}_f$. Its informational model that describes the data structure being manipulated is given by the type of the attribute *status* associated with each task. Its lifecycle model that provides information on tasks and their sequencing (coordination) is given by the GMWf

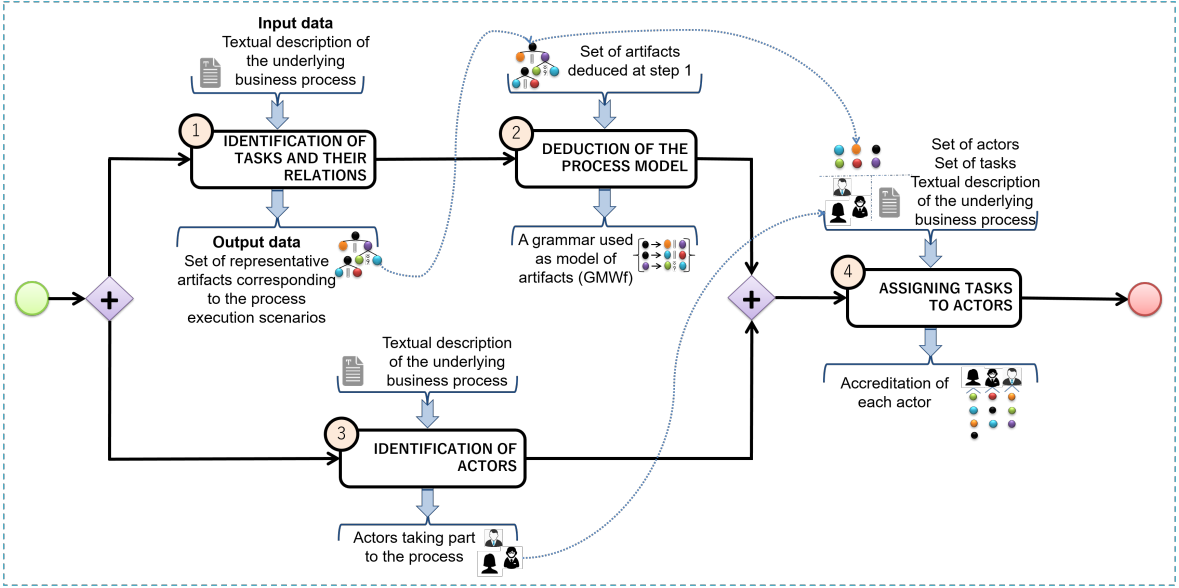

Figure 3: The main activities that are carried out when modelling a process using LSAWfP.

$\mathbb{G}$ of $\mathbb{W}_f$. Thus, we can conclude that LSAWfP has the major expected characteristics of a workflow language according to [1].

The GMWf effectively allows the designers to specify all the basic control flows (sequential, parallel, alternative and iterative) which can be found in traditional workflow languages. Figure 4 gives for each type of basic control flow its BPMN notation and the corresponding notations (artifact and associated productions) in LSAWfP as described below:

- The sequential flow between two tasks A and B can be expressed either by a production p of the form $p : A \rightarrow B$, or by a production q of the form $q : S \rightarrow A\,;B$ in which S is a (re)structuring symbol (see Figure 4(a));

- The parallel flow between two tasks A and B is expressed using a production p of the form $p : S \rightarrow A \parallel B$ (see Figure 4(b));

- The alternative flow (choice) between two tasks $A1$ and $A2$ is expressed using two productions $p1$ and $p2$ such that $p1 : S \rightarrow A1$ and $p2 : S \rightarrow A2$; S is a (re)structuring symbol expressing the fact that after "execution" of S, one must execute either task $A1$ or task $A2$ (see Figure 4(c));

- Iterative routing (repetition) is expressed using recursive symbols. Thus the productions $p1 : A \rightarrow B$, $p2 : B \rightarrow C$ and $p3 : B \rightarrow A$ express a potentially (transitive) iterative flow on the task A (see Figure 4(d)); $P_7 : \ E1 \rightarrow E1$ in the running example also expresses a direct iterative flow on $E1$ (see Figure 2).

As defined in this paper, LSAWfP (like traditional workflow languages) is only interested in modelling the inherent characteristics of administrative processes: i.e. tasks and their scheduling, data produced and consumed by tasks, actors in charge of executing the tasks and their roles. Aspects related to the execution of processes are not taken into account. LSAWfP therefore differs from the majority of artifact-centric languages. For example, the proclet approach [7] requires the designer to take into account aspects related to communication (ports, channels, etc.) between the entities in

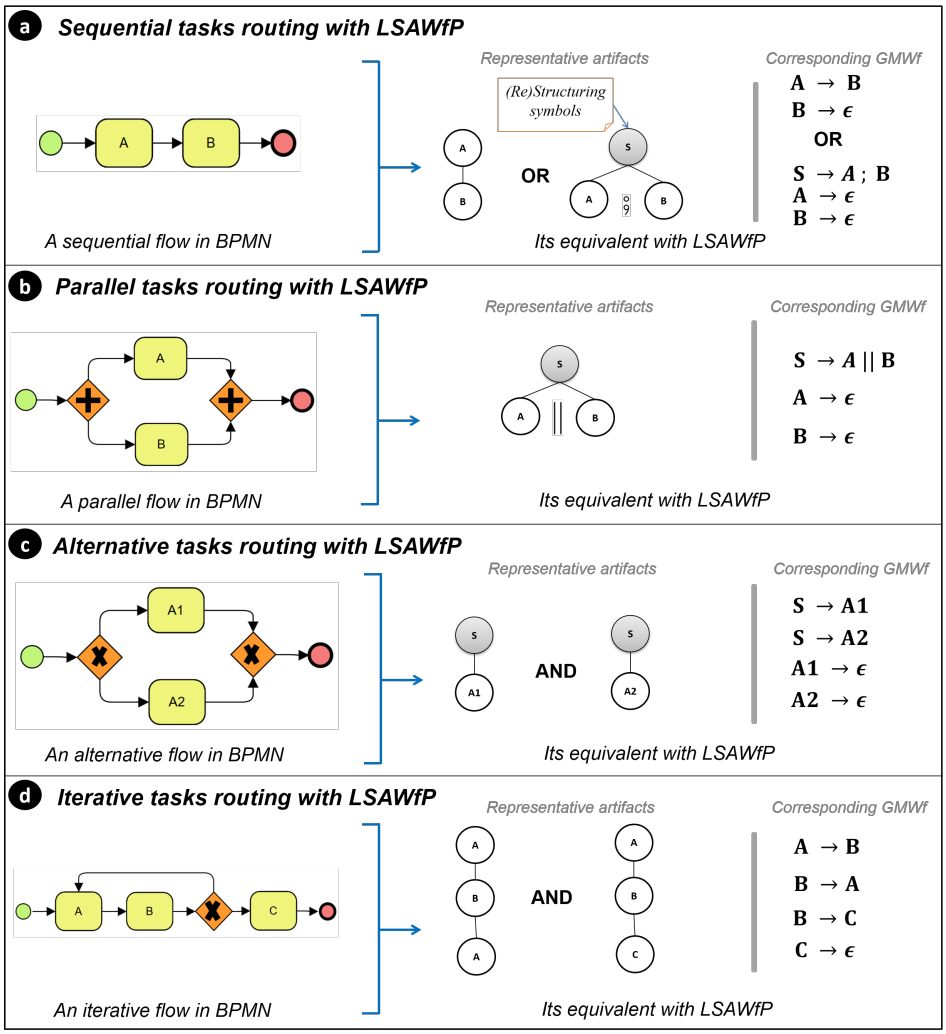

Figure 4: Illustrating basic control flows with LSAWfP.

charge of executing tasks (the proclets). The approach in [19] imposes to express also artifacts' locations (artifacts can be mobile or not). The expressiveness of LSAWfP does not allow the designer to specify such aspects.

The language LSAWfP has several interesting features in addition to its expressiveness; in particular:

– Its ability to represent scenarios using simple graphs (annotated trees) where existing languages use graphical formalisms (arbitrary graphs) that are more complex to implement;

– Its semi-declarative approach which would like the designer to describe the scheduling of a subset of tasks in a given production without expressing how the whole set of tasks is performed procedurally;

– Its usage of accreditations to model organizational aspects of process, making it particularly suitable for the modelling of administrative processes;

– Its modular approach using the scenario as the modelling unit;

– Its solid mathematical foundation mainly made up of a grammatical model that can be studied formally in the same way as Petri nets, while benefiting from the executable character that is recognized in such a tool.

In Table 2, we make a preliminary comparative study of LSAWfP with some of the workflow languages (those that we find best related to LSAWfP) presented in section 2.2. The criteria we have retained are as follows:

– The mathematical foundation: we specify the mathematical tool that serves as a basis for the language.

– The modelling paradigm: we clarify whether the language advocates a *procedural* approach in which one says how the process is carried out or whether it advocates a *declarative* approach in which, tasks are described simply and their sequences are specified using rules.

– The modelling formalism: does language propose a graphic formalism (*graph-based*), a textual formalism using rules (*rule-based*), or both ?

– The modelling unit: either *process* to refer to the fact that the whole process is modelled at once, or *user perception* to refer to the fact that the process is cut and modelled according to the perceptions of each actor on it, or *scenario* to refer to the fact that each scenario of the process is modelled independently of the others.

Table 2: A comparison of LSAWfP with other workflow languages.

Language	**Mathematical foundation**	**Modelling paradigm**	**Modelling formalism**	**Modelling unit**	**Highlighted conceptual model**	**Has an implementation ?**	**Is execution-context independent ?**
LSAWfP	Attributed Grammars	Semi-procedural (when designing scenario trees) and semi-declarative (when deriving rules for the GMWf)	Graph-based (annotated trees) and rule-based (the GMWf)	Scenario	All the models	No	Yes
BPMN [2]	Arbitrary Directed Graphs	Procedural	Graph-based	Process	Organizational and lifecycle models, but studies extending them to other processes perspectives do exist	Yes	Yes
YAWL [1]	Petri Nets	Procedural	Graph-based	Process	Lifecycle models, but studies extending them to other processes perspectives do exist	Yes	Yes
AWGAG [24]	Guarded Attributed Grammars	Declarative	Rule-based	User perception (modelling of active workspaces)	Informational and lifecycle models	No	No
Proclets [7]	Petri Nets	Procedural	Graph-based	User perception (modelling of proclets' classes)	Informational and lifecycle models	Not sure	No

– The highlighted conceptual model: we specify which of the informational, organizational and lifecycle models is emphasized by the considered language.

– The implementation: here we specify whether the language has an implementation or not.

– The independence regarding an execution technique: we specify if the language does not take into account aspects related to the technique and execution environment of the modelled processes.

These comparison criteria are not exhaustive and it is necessary to conduct a further study in order to better compare these languages. It would also be more appropriate to consider several other languages in this comparative study.

The LSAWfP language as presented here, is not perfect. The first criticisms we can make are the following:

– The organizational and informational models of LSAWfP formalized in this paper are quite simple. They are sufficient for the basic work carried out here to present the main concepts taken into account by this new language. However, it would be wise in further work, to better study organizations with administrative processes to improve these models.

– The expressiveness of LSAWfP has been analysed on basic routings and its applicability has been established on some examples of processes among which, the one described in this paper. Since LSAWfP also has a solid mathematical foundation, we have no doubt about its applicability in practice. However, it is essential to evaluate it in the modelling of larger processes in terms of number of actors, tasks, events, distribution (geographical distribution), etc. to definitively validate it.

4 Ongoing Work on LSAWfP

There is still a lot of work to be done to refine our models and achieve our goal of producing a complete workflow management infrastructure (a complete and solid workflow language, tools to assist in the design and validation of processes, a workflow execution environment, etc.). In this section, we present some of the work being currently done on LSAWfP.

LSAWfP and workflow patterns: one avenue we are currently exploring is that of measuring the expressiveness of LSAWfP in relation to workflow patterns [39]. This will allow us to characterize precisely the class of processes (beyond administrative processes) that this language can facilitate the modelling. To conduct this study, it is necessary to study in detail the different workflow patterns proposed in [39], then find examples of processes highlighting these patterns and finally, find out how LSAWfP can help to model these processes.

Towards a blockchain-like artifact-centric model of processes design and distributed execution based on cooperative edition of a mobile artifact: we are also working to produce an artifact-centric model of business process management. In this model inspired by the work of Badouel et al. on cooperative editing [40, 41, 42, 43, 44, 45], the process tasks are executed by the various actors with the help of software agents that they pilot. These software agents are autonomous, reactive and communicate in peer to peer mode by exchanging an artifact (considered as "mobile") edited cooperatively. This mobile artifact is an annotated tree that represents the execution status of the process at each moment. For this purpose, it contains information on the tasks already executed, on the data produced during these executions and on the tasks ready to be executed.

When the mobile artifact is received at a given execution site, the local agent executes an update protocol whose purpose is to reveal the tasks ready to be executed locally by the local actor. The

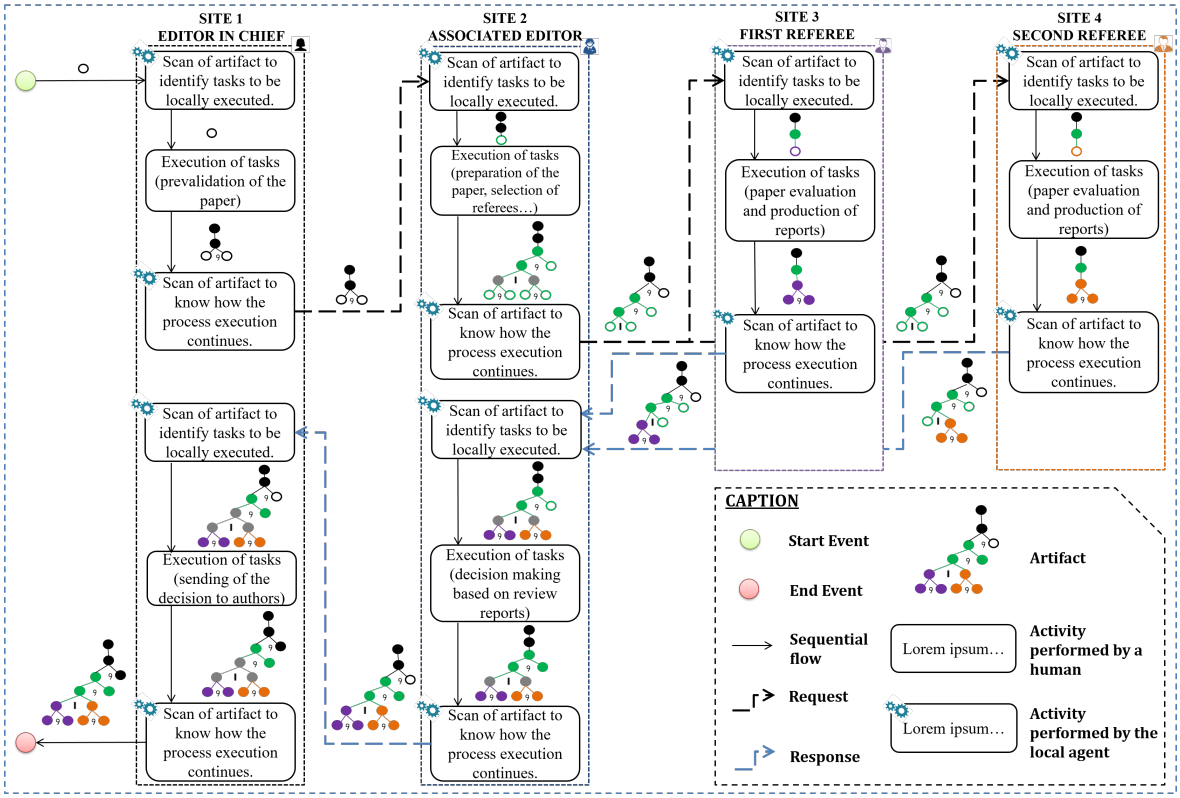

Figure 5: An overview of the artifact-centric execution of the peer-review process.

execution of the tasks by the local actor is done using a specialized editor and can be assimilated to the edition of a structured document since its actions cause the received mobile artifact (the tree) to be updated, by expanding some of its leaf nodes into sub-trees and by assigning values to the "status" attributes of some other nodes. When all the tasks ready to be locally executed have been executed, the artifact is sent to other agents for further execution of the process if necessary.

To run the peer-review process described in section 2.3 with the artifact-centric model being built, four agents controlled by four actors (the EC, the AE, the $R1$ and the R agents) will be deployed. Figure 5 sketches an overview of exchanges that can take place between those four agents. The scenario presented there corresponds to the nominal one in which the paper is pre-validated by the EC and therefore, is analysed by a peer review committee. The artifact-centric execution is triggered on the EC's site by introducing (in this site) an artifact reduced to its root node. During its transit through the system, this artifact grows. Note that there may be situations where multiple copies of the artifact are updated in parallel; this is notably the case when they are present on site 3 (first referee) and 4 (second referee).

5 Conclusion

In this paper, we have proposed a new workflow language called LSAWfP which allows, through a simple grammar-based formalism, to specify administrative business processes. Like any traditional workflow language, LSAWfP allows to specify basic flows (sequential, parallel, alternative and iterative) that are generally found in workflow models; particularly, it focuses on the modelling of each of the process scenarios using an artifact. Moreover, LSAWfP allows to model the main characteris-

tics of business processes (their lifecycle, their informational and their organizational aspects); it also allows to address certain security aspects of administrative workflows. In fact, LSAWfP allows the workflow models designers, to simply express each actor's accreditations for each task in a process, by the means of a formalism inspired by that used in UNIX-like operating systems for the expression of users' rights. We also presented some of the work associated with LSAWfP that are currently in progress.

It would certainly be easier to handle LSAWfP if we had a (graphical) tool to assist in the design and validation of its instances. Such a tool could be built as a classical software engineering workbench through which the user would specify his processes by drawing trees and modifying the properties of their nodes, and then export valid specifications into dedicated formats after verification by simulation using a tool integrated into the workbench. Moreover, it seems equally important to more precisely describe the model for executing business processes specified in LSAWfP and briefly presented in section 4. In our opinion, this is just a few of the many studies that must be carried out following the one presented in this paper.

Authors' Information

– **Milliam Maxime Zekeng Ndadji** is a PhD student in Computer Science at the University of Dschang (Cameroon), holding a Master of Science and a Bachelor of Science in Mathematics and Computer Science at the same university. His work focuses on the design of systems to support collaboration using formal tools such as grammars and automata.

– **Maurice Tchoupé Tchendji** holds a PhD in Software Engineering obtained in co-supervision at the Universities of Yaoundé I (Cameroon) and Rennes I (France); he is currently a senior lecturer and researcher at the University of Dschang (Cameroon). His work focuses on collaborative systems, XML databases, distributed systems, ad-hoc networks, improving the user experience through software localisation and machine learning.

– **Clémentin Tayou Djamegni** obtained the Third Cycle Doctorate and the State Doctorate at the University of Yaounde I (Cameroon) in 1997 and 2005 respectively. He has been a senior lecturer and researcher at the University of Dschang (Cameroon) since 1996. His main interests concern algorithm parallelization, regular networks, concept formal analysis, distributed algorithms, distributed systems and the boolean satisfiability problem.

– **Didier Parigot** holds a PhD from the University of Paris-Sud (now Paris-Saclay, France) and is currently senior researcher on programming language at INRIA (Sophia Antipolis, France). He is interested in formal languages, distributed systems, service-oriented architectures, peer-to-peer computing, component-based software engineering, domain-specific languages and generative programming.

Authors' Contributions

– **Milliam Maxime Zekeng Ndadji** suggested the idea of producing a workflow language based on an idea to produce a decentralised model of structured cooperative editing put forward by Maurice Tchoupé Tchendji and Didier Parigot. He then participated in the conception and formalisation of the said language, in the study of the latter with the help of several examples, in the writing and the proofreading of this paper.

– **Maurice Tchoupé Tchendji** is at the root of the work from which the one in this paper is derived. He participated in the formalisation of the language presented here, in its illustration, in the writing and the proofreading of this paper.

– **Clémentin Tayou Djamegni** is the co-supervisor of this work; he validated the various proposed mathematical tools, validated the examples that have been developed and contributed to the proofreading of this paper.

– **Didier Parigot** is (with Maurice Tchoupé Tchendji) at the base of the work that gave rise to this one; he is also co-supervisor of the work presented here. He has therefore validated the proposed mathematical tools and validated the examples that have been developed.

Competing Interests

The authors declare that they have no competing interests.

Funding

No funding was received for this project.

References

[1] W. M. Van der Aalst, "Business process management: a comprehensive survey," *ISRN Software Engineering*, vol. 2013, 2013.

[2] B. P. Model, "Notation (BPMN) version 2.0," *OMG Specification, Object Management Group*, pp. 22–31, 2011.

[3] W. M. Van Der Aalst and A. H. Ter Hofstede, "Yawl: yet another workflow language," *Information systems*, vol. 30, no. 4, pp. 245–275, 2005.

[4] E. Börger, "Approaches to modeling business processes: a critical analysis of bpmn, workflow patterns and yawl," *Software & Systems Modeling*, vol. 11, no. 3, pp. 305–318, 2012.

[5] M. Zur Muehlen and J. Recker, "How much language is enough? theoretical and practical use of the business process modeling notation," in *Seminal Contributions to Information Systems Engineering*, pp. 429–443, Springer, 2013.

[6] A. Nigam and N. S. Caswell, "Business artifacts: An approach to operational specification," *IBM Systems Journal*, vol. 42, no. 3, pp. 428–445, 2003.

[7] W. M. Van Der Aalst, P. Barthelmess, C. A. Ellis, and J. Wainer, "Proclets: A framework for lightweight interacting workflow processes," *International Journal of Cooperative Information Systems*, vol. 10, no. 04, pp. 443–481, 2001.

[8] W. M. Van Der Aalst, R. Mans, and N. C. Russell, "Workflow support using proclets: Divide, interact, and conquer.," *IEEE Data Eng. Bull.*, vol. 32, no. 3, pp. 16–22, 2009.

[9] S. McCready, "There is more than one Kind of Workflow Software," *Computerworld*, vol. 2, 1992.

[10] W. M. P. V. D. Aalst, "The Application of Petri Nets to Workflow Management," *Journal of Circuits, Systems, and Computers*, vol. 8, no. 1, pp. 21–66, 1998.

[11] M. Divitini, C. Hanachi, and C. Sibertin-Blanc, "Inter-organizational workflows for enterprise coordination," in *Coordination of Internet agents*, pp. 369–398, Springer, 2001.

[12] R. Lu and S. Sadiq, "A survey of comparative business process modeling approaches," in *International Conference on Business Information Systems*, pp. 82–94, Springer, 2007.

[13] G. Knolmayer, R. Endl, and M. Pfahrer, "Modeling processes and workflows by business rules," in *Business Process Management*, pp. 16–29, Springer, 2000.

[14] M. Reichert, S. Rinderle, and P. Dadam, "Adept workflow management system," in *International Conference on Business Process Management*, pp. 370–379, Springer, 2003.

[15] E. T. Almeida, J. E. Luntz, and D. M. Tilbury, "Modular finite state machines implemented as event-condition-action systems," *IFAC Proceedings Volumes*, vol. 38, no. 1, pp. 373–378, 2005.

[16] M. Abi Assaf, "Towards an integration system for artifact-centric processes," in *Proceedings of the 2016 on SIGMOD'16 PhD Symposium*, pp. 2–6, ACM, 2016.

[17] A. Deutsch, R. Hull, and V. Vianu, "Automatic verification of database-centric systems," *ACM SIGMOD Record*, vol. 43, no. 3, pp. 5–17, 2014.

[18] R. Hull, N. C. Narendra, and A. Nigam, "Facilitating workflow interoperation using artifact-centric hubs," in *Service-Oriented Computing*, pp. 1–18, Springer, 2009.

[19] N. Lohmann and K. Wolf, "Artifact-centric choreographies," in *International Conference on Service-Oriented Computing*, pp. 32–46, Springer, 2010.

[20] M. A. Assaf, Y. Badr, and Y. Amghar, "A continuous query language for stream-based artifacts," in *International Conference on Database and Expert Systems Applications*, pp. 80–89, Springer, 2017.

[21] M. A. Assaf, Y. Badr, H. El Khoury, and K. Barbar, "Generating database schemas from business artifact models," *I.J. Information Technology and Computer Science*, vol. 2, pp. 10–17, 2018.

[22] D. Boaz, L. Limonad, and M. Gupta, "Bizartifact: Artifact-centric business process management, june 2013," 2013.

[23] E. Badouel, L. Hélouët, G. E. Kouamou, and C. Morvan, "A Grammatical Approach to Data-centric Case Management in a Distributed Collaborative Environment," *CoRR*, vol. abs/1405.3223, 2014.

[24] E. Badouel, L. Hélouët, G.-E. Kouamou, C. Morvan, and N. R. Fondze Jr, "Active workspaces: distributed collaborative systems based on guarded attribute grammars," *ACM SIGAPP Applied Computing Review*, vol. 15, no. 3, pp. 6–34, 2015.

[25] M. M. Zekeng Ndadji, M. Tchoupé Tchendji, C. Tayou Djamegni, and D. Parigot, "A language for the specification of administrative workflow processes with emphasis on actors' views," in *Gervasi O. et al. (eds) Computational Science and Its Applications – ICCSA 2020. ICCSA 2020. Lecture Notes in Computer Science*, vol. 12254, pp. 231–245, Springer, 2020.

[26] M. M. Zekeng Ndadji, M. Tchoupé Tchendji, C. Tayou Djamegni, and D. Parigot, "A grammatical model for the specification of administrative workflow using scenario as modelling unit," in *Florez H., Misra S. (eds) Applied Informatics. ICAI 2020. Communications in Computer and Information Science*, vol. 1277, pp. 131–145, Springer, 2020.

[27] G. J. Fakas and B. Karakostas, "A Peer to Peer (P2P) Architecture for Dynamic Workflow Management," *Information & Software Technology*, vol. 46, no. 6, pp. 423–431, 2004.

[28] J. Yan, Y. Yang, and G. K. Raikundalia, "SwinDeW-a P2P-Based Decentralized Workflow Management System," *IEEE Trans. Systems, Man, and Cybernetics, Part A*, vol. 36, no. 5, pp. 922–935, 2006.

[29] H. Huang, R. Peng, Z. Feng, and M. Zhang, "A cloud workflow modeling framework using extended proclets," in *Asia-Pacific Conference on Business Process Management*, pp. 19–34, Springer, 2015.

[30] L. García-Bañuelos, A. Ponomarev, M. Dumas, and I. Weber, "Optimized execution of business processes on blockchain," in *International Conference on Business Process Management*, pp. 130–146, Springer, 2017.

[31] B. Carminati, C. Rondanini, and E. Ferrari, "Confidential business process execution on blockchain," in *2018 IEEE International Conference on Web Services (ICWS)*, pp. 58–65, IEEE, 2018.

[32] C. Sturm, J. Szalanczi, S. Schönig, and S. Jablonski, "A lean architecture for blockchain based decentralized process execution," in *International Conference on Business Process Management*, pp. 361–373, Springer, 2018.

[33] G. Falazi, M. Hahn, U. Breitenbücher, and F. Leymann, "Modeling and execution of blockchain-aware business processes," *SICS Software-Intensive Cyber-Physical Systems*, vol. 34, no. 2-3, pp. 105–116, 2019.

[34] O. López-Pintado, L. García-Bañuelos, M. Dumas, I. Weber, and A. Ponomarev, "Caterpillar: a business process execution engine on the ethereum blockchain," *Software: Practice and Experience*, vol. 49, no. 7, pp. 1162–1193, 2019.

[35] C. Sturm, J. Scalanczi, S. Schönig, and S. Jablonski, "A blockchain-based and resource-aware process execution engine," *Future Generation Computer Systems*, vol. 100, pp. 19–34, 2019.

[36] C. Di Ciccio, A. Cecconi, M. Dumas, L. García-Bañuelos, O. López-Pintado, Q. Lu, J. Mendling, A. Ponomarev, A. B. Tran, and I. Weber, "Blockchain support for collaborative business processes," *Informatik Spektrum*, vol. 42, no. 3, pp. 182–190, 2019.

[37] R. Nsaibirni, *A Guarded Attribute Grammar Based Model for User Centered, Distributed, and Collaborative Case Management Case of the Disease Surveillance Process*. Theses, Université de Yaoundé I, 2019.

[38] M. Tchoupé Tchendji and J. Ngoufo Tagueu, "A publish/subscribe approach for implementing GAG's distributed collaborative business processes with high data availability," in *CARI 2020 - African Conference on Research in Computer Science and Applied Mathematics*, (Thiès, Senegal), 2020.

[39] W. M. Van Der Aalst and A. H. ter Hofstede, "Workflow patterns put into context," *Software & Systems Modeling*, vol. 11, no. 3, pp. 319–323, 2012.

[40] E. Badouel and M. T. Tchendji, "Merging Hierarchically-Structured Documents in Workflow Systems," *Electronic Notes in Theoretical Computer Science*, vol. 203, no. 5, pp. 3–24, 2008.

[41] M. Tchoupé Tchendji, R. D. Djeumen, and M. T. Atemkeng, "A Stable and Consistent Document Model Suitable for Asynchronous Cooperative Edition," *Journal of Computer and Communications*, vol. 5, no. 08, p. 69, 2017.

[42] M. Tchoupé Tchendji and M. M. Zekeng Ndadji, "Réconciliation par consensus des mises à jour des répliques partielles d'un document structuré," in *CARI 2016 Proceedings*, vol. 1, pp. 84–96, 2016.

[43] M. Tchoupé Tchendji and M. M. Zekeng Ndadji, "Tree Automata for Extracting Consensus from Partial Replicas of a Structured Document," *Journal of Software Engineering and Applications*, vol. 10, no. 05, p. 432, 2017.

[44] M. M. Zekeng Ndadji and M. Tchoupé Tchendji, "A Software Architecture for Centralized Management of Structured Documents in a Cooperative Editing Workflow," in *Innovation and Interdisciplinary Solutions for Underserved Areas*, pp. 279–291, Springer, 2018.

[45] M. M. Zekeng Ndadji, M. T. Tchendji, and D. Parigot, "A Projection-Stable Grammatical Model to Specify Workflows for their P2P and Artifact-Centric Execution," in *CRI'2019 - Conférence de Recherche en Informatique*, (Yaoundé, Cameroon), 2019.