Flexible Formal Specifications to Design Robust Technology-Enhanced Learning Applications

Juan Felipe Calderon[1](✉) and Luis A. Rojas[2]

[1] Facultad de Ingeniería, Universidad Andrés Bello, Quillota, 980 Viña del Mar, Chile
juan.calderon@unab.cl

[2] Facultad de Ciencias Empresariales, Departamento de Ciencias de la Computación y Tecnologías de la Información, Universidad del Bío-Bío, Chillán, Chile
lurojas@ubiobio.cl

Abstract. In real-time systems with changes in specifications, resources status, and ambient conditions, the computational support requires adapting to new conditions to satisfy the goals defined for those applications. Technology-enhanced learning applications (TEL) can be formalized by Educational Modelling Languages (EML), which provides a mechanism to design, deploy, and execute learning activities providing pedagogical flexibility. However, this flexibility is focused on design time and not in execution. Consequently, compliance satisfaction is a challenge when flexibility in workflows is provided. In addition, deadlock-freeness and reachability are critical properties in learning design execution of applications because learning objectives must be achieved by students without avoidable impediments provided by the execution of learning applications and their corresponding technical infrastructure. Currently, to provide flexibility to learning design scenarios, real-time flexibility and real-time compliance cannot be balanced since the improvement of flexibility mechanism affects compliance assessment, and vice-versa. The aim of this work is to explore real-time flexibility features in a workflow specification, supporting deadlock-freeness and reachability as compliance parameters during application in learning design scenarios. An extension of Petri-Net formalism was developed as a workflow specification. To validate this approach, a learning scenario with a set of test cases were formulated to define pedagogical and validation constraints. Results show that a learning scenario involving changes to the run-time can be successfully created, deployed, and executed. These changes can be based on properties that are intrinsic to the learning scenario, as well as on others that are related to the proposed workflow specification base Petri-net.

Keywords: flexible workflow · learning design · petri-net · model-checking

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/978-3-031-35927-9_2.

1 Introduction

Real-time systems and applications are conditioned to heterogeneous hardware stakeholders, ambient factors, and subjacent processes. Consequently, these systems may require adapting to the new working conditions to satisfy, complete or partially, predefined goals [1]. In this line, Technology-enhanced learning (TEL) applications are real-time applications: they are deployed on portable and interoperable devices; they are designed for a wide technology-level range of users; they are ambient-pervasive according to indoor and/or outdoor settings; and their aim is to facilitate how people learn, according to pre-defined learning results. This complexity must be tackled by an adequate heterogeneity management to a fulfilment of the expected learning results by the targeted users, avoiding overloading them. In this line, heterogeneity management can be supported by formalisms to provide mechanisms of runtime verification and validation. In TEL field, a relevant and current concept is Learning Design, as an approach to provide learning technological applications with a correct pedagogical design [2–4].Currently, there are several learning design tools focused on content delivery, students' assessment, pedagogical planning, and authoring [5, 6] Particularly regarding to pedagogical planning and authoring, heterogeneity management must be aligned to how Learning Design is formally represented [7, 8]. Educational Modelling Languages (EML) [9] provides formalisms to Learning Design approach. Any EML implementation must provide not only formalization, but pedagogical flexibility, personalization, interoperability, sustainability, and reusability. Most-known EML implementation is IMS-LD [10], an XML-based language for TEL applications formalization providing a domain-specific metamodel with three complexity levels:

- Level A – definition of core entities: pedagogical method, plays, acts, roles, learning activities, environment setting and status, etc.
- Level B – monitoring over level A status.
- Level C – notifications over level B, related to monitoring process in that level.

As seen, IMS-LD level B This level provides mechanisms to create more heterogenous and complex TEL applications using state variables, pre-defined properties, and flexibility conditions [11]. Then, TEL applications can be created with a flexible learning flow, i.e., with a sequence of activities and tasks to achieve learning results, introducing new tasks, rules, or pedagogical resources. Therefore, all possible modifications in learning flow must be defined as a pool, before runtime execution with students, without real-time interaction by a user (e.g., lecturer or teacher). To solve these issues, [12] propose an architecture to orchestrate external software components into an IMS-LD implementation using human or automatic agents (i.e., a virtual teacher), as mediators. Other approaches to solve flexibility issues are related to how provide an adequate learning flow to improve learning [13], how flexibility can be implemented in constraints and variable geolocations and places (e.g., [14]) and flexibility in authoring when task of learning flow is designed (e.g., [15]). These approaches extend IMS-LD, providing events and exception support, creation and insertion of new tasks and distributed data exchange, but all of them pre-defined. Therefore, an unexplored issue is how an EML formalism (e.g., IMS-LD) can specify flexible Learning Designs, considering heterogeneity management and real-time supporting on pedagogical planning changes [7]. As seen, TEL applications are limited to solve issues in a design stage, because flexibility is provided by a pool of rules described by the corresponding EML implementation.

Related to heterogeneity management, an issue is how variated and spare components and data resources support a right learning results achievement by learners. As seen, [12] focuses on flexible orchestration to satisfying complex requirements in collaborative learning, using some distributed components and resources to satisfy pedagogical requirements. This approach uses IMS-LD as specification formalism, extending it using service choreography with external resources to provide adaptation in content presentation to learners. Its aim is to provide robustness features related to group formation and external resources, but there is not flexibility in task descriptions or exception handling.

1.1 Workflows and EML

The EML formalisms may be considered as an instance of workflow concept. A workflow is composed by a sequence of steps, where certain tasks are carried out using specific resources, to satisfy pre-defined goals [16] in a wide range of business domains [17]. Workflow executions are supported by its specification (i.e., a formal language), and an operative infrastructure over workflow management systems (WfMS). Previous research has looked at introducing changes without having to restart the entire system. These approaches are called flexible workflows [18]. Flexibility in this type of workflow can operate in three ways. First, the decision making is delayed when faced with an event (e.g., specifying a spot in the workflow definition to be filled, marked by a language sentence). Second, new tasks can be added or changed to build the workflow sequence immediately, both for current and future executions. Finally, the current model is ignored by skipping tasks and/or violating restrictions that were in place with the previous tasks and running sequence [19].

As aforementioned, TEL applications may need flexibility and heterogeneity management in their operation. Regarding heterogeneity, a TEL application can be deployed over various kind of systems: cloud servers, mobile devices, and recently, IoT devices [20]. Another method is to make nodes available for playing the role of both data and control servers, as well as executors of the workflow. This allows distributed control of the process, in addition to the migration of the execution of processes between nodes, enabling runtime flexibility [21]. The advantage of these systems is the autonomy they can provide each node for decision making. However, this autonomy can result in potential problems with syncing, reachability of all possible states in workflow, possible locks in tasks and resources access given the way they are set up as a distributed system, and temporal availability of resources [22].

1.2 Execution Control and Rules

In any TEL application specification, the mechanism of execution control needs to be defined. First, transition between tasks must be defined, generally embedded into a sequence of tasks. Then, traceability between technical requirements and expected learning results must be defined. This traceability is specified by rules, statements that define or constrain some aspects of a certain problem. Precisely, it is intended to assert domain structure to control the behavior of the learning flow. Examples of this kind of rules are: “each student must achieve a minimum score in a task to develop a next task”; “every student must work in a group”. Rules-based approach distinguishes between definitions, facts, and formal rules, such as constraints, derivation, or reaction rules [23]. These rules must be associated to expected learning results, because a traceability between pedagogical and technical requirements it is needed to get success in any TEL application. An issue in this traceability is how rules are formally described in a workflow specification language. If the rules are explicit, then the way the rules are incorporated in the definition will depend on the language, in which the learning design is defined if this is modelled by a workflow specification, and how is this specification is adopted by designers and instructors [5].On the other hand, when rules are implicit, these can be deduced from other elements in the specification, and it is not explicitly coded in the specification. This fact can be illustrated using the analogy with the process of putting on shoes and socks: a first step is to put on the socks, followed by the shoes; then, based on this sequence, the deduced rule is that the socks must be covered by the shoe. However, a rules deduction is not useful because it inhibits verification and validation processes with not clear sequence reference parameters.

1.3 Verification and Validation of Flexibility in Learning Designs

Flexibility enables to add new tasks to workflows, according to the state of resources, ambient, and other factors. Nonetheless, flexibility may add other issue: every change within the workflows should be validated against a set of goals and requirements [24]. Regarding TEL applications, compliance in flexibility and goals satisfaction are balanced depending on the context where the application is deployed and executed. This is needed to determine whether the learning results can be still fulfilled before making any kind of modification to guarantee the robustness of the TEL application. Then, specification flexibility and its verification in real time may not be trivial, depending on how the learning flow is represented. An issue is how to verify if changes in its specification are internally consistent with their other element in the same specification [22]. If these changes are only in the sequence of tasks, this process can be automated. On the other hand, this activity is manual, if there is a redefinition of tasks which requires time and effort. Therefore, a manual redefinition decreases performance of workflow execution [24]. From a technical point of view, for TEL applications verification process is critical with respect to how non-functional requirements are defined. A right definition has an impact in the whole performance in execution time and processing when a TEL application is executed in a real-time context, such as a presential classroom or remote synchronous activities. On the other hand, from a pedagogical point of view, verification and validation can be subordinated to how the learning design adapts its specifications (parameters, resources, state variable), as a medium to satisfy learning results.

Some examples in the literature attempt to incorporate the change within the workflow itself. One of the main attempts has been to integrate rules into workflows that explicitly represent the restrictions of the domain. This is a critical issue in learning design, because the definition of constraints is important to design flexible systems supporting unexpected events of the enactment of learning design scripts [14]. A first approach is to directly incorporate blocks of code that represent the rules in the code where the workflow is defined. However, this makes changes to the workflow difficult and therefore makes it difficult to maintain or to become inflexible [21]. The rules that can be defined using this approach must be simple (i.e., without using variables that represent the state of the workflow). These rules must also be local without including workflows with distributed elements [21]. One way of solving these problems is to take full advantage of using service-oriented architecture by explicitly decoupling the sequence of tasks in the workflow from the rules. For example, the rules are stored and performed in external rule engines, which are requested by the workflow using web services [23].This approach allows a posteriori flexibility as it defines spots, where the rules that are contained in a web service and remain decoupled from the workflow can be modified. A limitation of this approach is that as it does not allow changes to the running order when the workflow is being performed because such changes could affect the rules or require new rules.

Creating a workflow with the characteristics indicated above makes the process of verifying properties after changes more complex. According to the literature, the verification of properties in imperative workflows can be achieved independently using Petri-nets [25]. Petri-nets come from a language for mathematical modelling, which allows distributed systems to be described using a bipartite graph. Additionally, the elements of a Petri-net represent the semantics of the modelled process and allow properties, which are expected to be met during the execution of a workflow to be checked, such as deadlock-freeness, reachability of workflows expected states, safety, and liveness [16]. One disadvantage of separately verifying the properties of workflows and rules is that the compliance to these properties cannot be directly assessed. This issue emerges because there is dependency between workflow tasks and the rules that define the transition between workflow tasks. This assessment is required when making changes to the workflow that affect the running order or the definition of the set of rules because inconsistencies may appear due to the changes.

1.4 Modelling Workflows and Rules Using Petri-Nets

In the literature, an approach for modelling and formally defining workflows has been performed explicitly using Petri-nets. Nevertheless, there are approaches that consider the problem of flexible rules, although they do not consider the definition of a workflow. This problem is partially covered using ontology-based Petri-nets [26], defining an extra semantic layer, to provide knowledge over the specified Petri-net. In this case, new rules and axioms can be integrated to the system, with no reprogramming, such as the OPENET LD approach [26]. However, this leads to the following problems: (1) Complex rules (i.e., rules that are made up of other rules through disjunction and conjunction) can return different results for the same input; and (2) Given that Petri-nets use decision thresholds to make decisions, calibrating this type of system leads to a high overhead for incorporating flexibility.

Consequently, following research questions emerge: (1) Is it possible to incorporate complex rules into an imperative workflow so as to allow dynamic changes (during run-time), both to the properties that govern it as well as the order in which the tasks are defined and performed, guaranteeing the integrity of the rules that govern the problem's domain?, and (2) Is it possible to define a methodology for automatic property verification based on Petri-nets for the workflow proposed in the first research question?. To answer these research questions, this paper is structured as following: Sect. 2 presents Petri-nets as a modeling tool; Sect. 3 presents the proposed integration of workflow specification and rules, Sect. 4 presents the validation of this proposal, and Sect. 5 presents the conclusions about this work.

2 Petri-Nets as a Formalism for Modelling Workflows and Rules

In the literature, an approach for modelling and formally defining workflows has been performed explicitly using Petri-nets [25]. Using this concept, the definition of a workflow can be directly translated to a representation in Petri-net. The flexibility is incorporated by inserting snippets or blocks of code into a workflow specification, such as BPEL. These snippets can be directly translated to a certain pattern in Petri-net. Although this allows for some flexibility, it is only to a certain extent. The changes are limited to adding and modifying tasks, but do not allow rules that establish a transition from one task to another to be incorporated. Therefore, the validation of the changes is based on how these matches with the patterns in Petri-net. While the creation of these is well defined according to the methodology, this is not the case for a change in the rules that govern the workflow.

Nevertheless, there are approaches that consider the problem of flexible rules, although they do not consider the definition of a workflow. These allow a world to be represented where changes can be made to the context of the problem. However, this leads to the following problems:

- Complex rules (i.e., rules that are made up of other rules through disjunction and conjunction) can return different results for the same input.
- Given that Petri-nets use decision thresholds to make decisions, calibrating this type of system leads to a high overhead for incorporating flexibility.

3 Integration of Workflows and Rules

This study proposes the definition of a framework for integrating workflow and rules, with a methodology for verifying properties. An approach like that described in [27] is used to develop the integration between workflow and rules. With this approach, the following elements must be defined separately:

- The definition of the rules that govern the application's domain. These rules are defined as they are logic propositions, programed in a certain language. With this, two types of rules must be distinguished: those which are invariant to the application's state (i.e., that must always be met), and those which regulate the change from one state to another (i.e., those that are used to control the flow of the sequence of tasks).
- The definition of the workflow that represents the temporal sequence of the tasks to be performed.

3.1 Formalization

This section gives a formal account of workflows integrated with rule. Before defining our workflows, we define formally what we mean by a variable. We assume that a *variable* v is a mathematical object with an associated domain Dom(v). An *assignment* is a set of tuples(v,k), where v is a variable and k is a value in Dom(v). Given a set of variables, a *rule* is defined inductively as:

- v OP K is a rule when OP is in, v is in V and K is a value in v's domain.
- OR(R) where R is a finite set of rules. Intuitively OR is the Boolean OR operation.
- AND(R) where R is a finite set of rules. Intuitively AND is the Boolean OR operation.

The truth value of a rule r can be determined given an assignment *a* that assigns a value to every variable in V. We say that r evaluates to true in A if r reduces to true given the assignment A, and we say that r evaluates to false in A if r reduces to false given A. An inductive definition is straightforward, and we omit it here. For example rule AND ($v = k$, $u > m$) evaluates to true if A assigns k to v and A assigns a value to u that is greater than m.

A workflow integrated with rules W is defined as a tuple, where:

- is a finite set of variables.
- is an (initial) assignment of values to all of the variables in V.
- is a set of sentences. There are two classes of sentences: (a) assignment of values to variables, denoted by, where is in the domain of variable and (b), where is a value and is a distinguished constant which intuitively denotes a procedure call.
- is a set of tuples where are sentences and is a rule.

Intuitively, tuples model transitions of the workflow. Tuple represents the fact that after finishing the execution of the control will transfer to if takes the value True, and to otherwise. To model unconditional transitions, which do not involve a rule (i.e., go from one task to another regardless of the state), we can use an expression that is always true (e.g.) for, and

- is the initial sentence.
- is the final sentence.

3.2 Execution

Given a workflow integrated with variables, we now define the semantics of an execution. To this end, we start off by defining *instant descriptions* which are mathematical representations of snapshots of the state of execution. Then we define how execution can move from one instant description to another.

Formally, an instant description is a tuple, where is a sentence and is an assignment to all variables in V. In addition, we define the binary relation between two instant descriptions, such that intuitively means that is reachable from in one step of computation. Formally, if and only if:

1. When then a' is like a, but with the assignment of variable v replaced with value b. Thus, a' assigns all the variables in, in the same way as a, and assigns b as a value of v.
2. If then is like, with v assigned to some value of the domain of
3. for some and some, belongs to and evaluates to true in or belongs to and evaluates to false in.

An execution of is a sequence, where, and for all in.
An execution is successful if.

3.3 Formalization of Petri-Nets

A Petri-net P is defined as a tuple where:

- corresponds to the set of places
- corresponds to the set of transitions
- corresponds to a set of tuples of the form, where are places, while corresponds to a transition, which directly relates two places to each other, from to.
- :initial place.
- :set of final places of the Petri-net

3.4 Translation

Given a workflow integrated with rules, W, we define below a procedure for generating a Petri-Net

1. For each variable v, we define one place for each value in.
2. For each sentence, we define a place.
3. For each transition
   a. If is or, we define two places, and.
   b. If it corresponds to the assignment of a value to the variable, a transition, is defined, where corresponds to the index of the sentence.

4. Following this, a transformation pattern to Petri-net is applied to each sentence, resulting in a sub-Petri-net.
5. After this, each resulting sub-Petri-net is added iteratively, depending on the function of the associated transition. With this and depending on whether or not can be true or false, the place represented by each value of the evaluated variables is added to the corresponding entries to the sub-Petri-net that must be executed if is true or false.

An example of some of the transformation patterns are included below.:

- AND: a pattern that represents an AND conjunction is represented as a Petri-net with as many entry places as there are rules in the conjunction. Each of these points to a transition, which in turn is directed to an exit place. This also makes it possible to add other conjunctions and disjunctions as they are also established as rules.
- OR: a pattern that represents an OR disjunction is represented as a Petri-net with as many entry places and transitions as there are rules in the disjunction. Each entry place points to a transition, and each transition points to the same exit place. This also makes it possible to add other conjunctions and disjunctions as they are also established as rules.
- Comparison = (equality): is represented as a Petri-net that has several entry places: one that corresponds to the value of the variable that is being evaluated in the comparison (i.e., the answer to the rule is True), others that correspond to the values that do not correspond to those that were evaluated (i.e., the answer to the rule is False), as well as another place that corresponds to the previous task that comes before the evaluation of the comparison. There are two exit places. The first is one that connects the place corresponding to the value of the variable that returns True with the place of the previous task using an AND. The other connects the places corresponding to the values of the variable that return False with the place of the previous task using an AND. Both have an exit place that connects with the sub-Petri-net that represents the relevant task to be performed once the condition has been evaluated, regardless of whether it is True or False.

Consequently, this work proposes the definition of a specification for flexible workflows, integrating domain rules with a technique for automated compliance assessment, to satisfy performance, deadlock-freeness, and reachability in workflow execution. In the field of learning design, rules specify corresponding pedagogical constraints; also, workflow defines the learning flow proposed by a TEL application as an instance of a learning design specification. Therefore, these elements must be defined independently and consequently must be integrated in a common specification:

The definition of the rules that govern the learning design instance. Two types of rules must be distinguished: those which are invariant to the application's state (i.e., that must always be met), and those which regulate the change from one state to another (i.e., that are used to control the flow of the sequence of tasks).

The definition of the workflow that represents the temporal sequence of the tasks in the pedagogical plan.

In Fig. 1 is presented an example of this idea. Using Petri-net notation, a small learning scenario is presented: an application for English pronunciation learning needs to evaluate the correction in pronunciation in a certain word and the rightness in the word choosing process. These events in learning process, signed as “validatePronunciation”, and “comparison”, correspond to the combinate use of a consume of an external service of pronunciation evaluation, called in transition between OR and Comparison stages, and an internal decision process implementation (OR, Comparison, & AND stages). In the first case, a “Decision as-a-service” approach is followed, similar to Hasić, et al. (2020) work. In the second case, the Petri-net design uses well-known patterns to represent corresponding explicit and implicit pedagogical constraints. In this example, the changes are limited to adding and modifying tasks, but do not allow pedagogical constraints to be incorporated in the workflow as a mechanism to establish a transition from one task to another. Therefore, the validation of the changes is based on how these constraints match with the patterns in Petri-net.

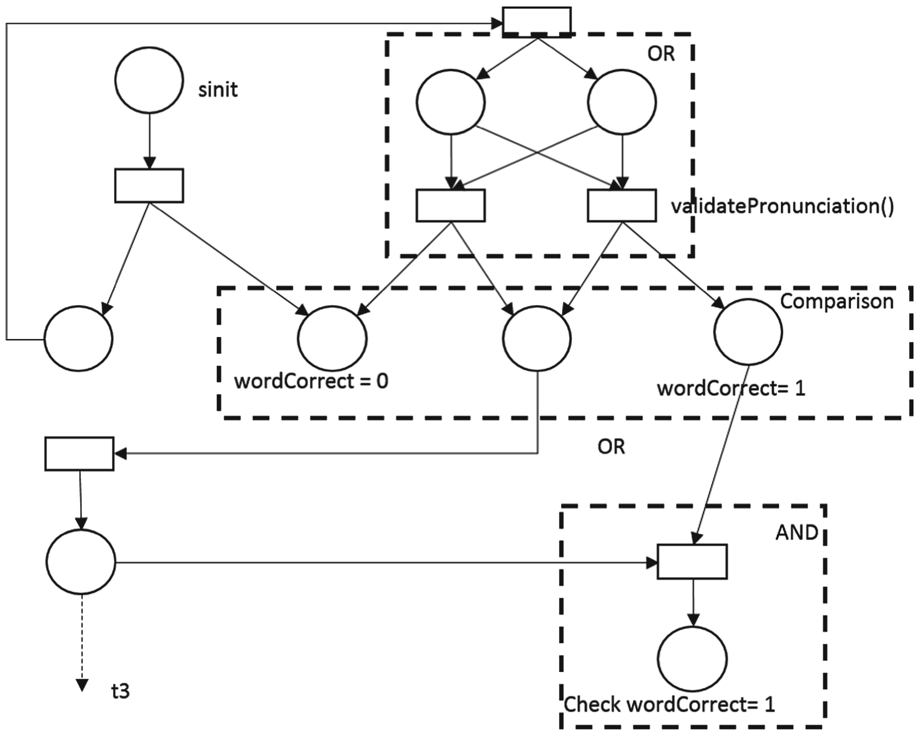

Fig. 1. Example of workflow specification translation into a Petri Net.

Context: A student must correctly pronounce a word. They have three opportunities to do so.

Rules (in natural language)
R1: The word must be correct to advance to the next activity
R2: There are three opportunities to correctly pronounce a word
Variables (V) and their initial configuration ()
correctWord = 0, attempts = 0;

Formalized rules
: correctWord = 1
: OR(attempts = 0, attempts = 1, attempts =2)

Tasks
t0: sinit
t1: isCorrect := 0;
t2: isCorrect := validatePronunciation()
t3: attempts = attempts + 1
t4: send

Transition function (T)
<t0, {1=1},t1,t1>
<t1,{1=1}t2,t2>
<t2,r1,t4,t3>
<t3,r2,t2,t4>

Goals (G)
<isCorrect = 1, t4>

3.5 Implementing the Formalization

BPEL is used to define the workflow to implement the methodology. BPEL was chosen as the language for representing the workflows because it is considered the industry standard and because it fully integrates into a service architecture. Furthermore, there are standardized methodologies for BPEL, as well as automatic translations to representations in Petri-nets. More specifically, the following tasks must be carried out:

- Define a WSDL file that represents the rules as a service. In this file, each rule is used through an entry-point. With this, the coding of the rule remains independent from the BPEL.
- Define synchronous calls to service in the BPEL that represents the workflow. This service represents the rules using the invoke and receive BPEL commands. The invoke command will reference the corresponding rule (using the name of the corresponding

entry-point) and the receive command will save the result of the request (true or false) for use controlling the flow of the application.

The BPEL2oWFN tool is used to translate the workflow to Petri-nets (Lohmann, 2007). This allows the translation to multiple representations of Petri-nets, to which the formalization presented above adjusts. To integrate the resulting Petri-nets, of both the workflow and the rules, a semi-automatic methodology was used to recognize the places and transitions in Petri-nets, based on the patterns used in the translation of BPEL2oWFN.

With this it is possible to find the points where the requests are made to the service representing the rules. Here, the initial and final points of the Petri-nets that represent each rule are inserted.

3.6 Validation and Correction

A model-checker for Petri-nets must be used for verifying the properties. The chosen model-checker is LoLA2 [28], which allows properties that are intrinsic to the Petri-nets to be examined, as well as ad-hoc properties that are expected of the domain (i.e., adherence to the proposed rules). These properties must be analyzed every time there is a change to the application, whether it be to the workflow or the rules. The properties that were analyzed are the following:

- Deadlock-freeness analysis: checks whether there are any possible configurations of the Petri-net that led to deadlock. It does not require parameterization.
- Reachability analysis: checks that each of the expected states is reached (i.e., the goals defined in the workflow). It requires the specification of the test input sets.
- Safety-Liveness analysis: Determines the temporal logics according to the sequence of the workflow. To be evaluated, the valid sequence of steps in the workflow is determined, before determining whether the rules are adhered to. To follow the indicated flow, the proposed rules must also be followed. Therefore, the introduction of valid inputs (i.e., the evaluation of reachability) must be analyzed together. The input model consists of the following elements:

  o Petri-net with model to be tested
  o Valid and invalid inputs for each rule
  o Expected states (post-conditions or goals)
  o Valid sequences within the context, written in Linear Temporary Logic – LTL –. The model-checker generates a combination of possible sequences (not necessarily feasible and/or consistent with the rules), with the aim of testing the Petri-net against boundary conditions.

Given that the Safety-liveness evaluation includes the other properties, the resulting output from the verification is the following:

- Safety-liveness evaluation indicates points in the Petri-net where all of the possible combinations of sequences fail.
- Reachability evaluation: indicates whether all the expected states were reached.
- Deadlock-freeness evaluation: indicating the points in the Petri-net where deadlocks occur.

4 Validating the Proposal

An extension of the scenario presented by [29] was used to validate the proposal presented in this study. This scenario consists of an application for learning a foreign language, based on single-display groupware and collaborative learning. In the original application, three students work in front of a shared screen, interacting with the application using headsets. The application allows the students to carry out activities which develop their grammar, vocabulary, listening skills and pronunciation. Automatic feedback is given by the system using a speech recognition engine and speech synthesizer. Both the use of multimedia in this scenario (such as the speech engines), as well as the collaborative learning dynamic, are of interest to this study because, based on these, new requirements can be developed. With these new requirements it is possible to formulate new task definitions and make changes to those that already exist. This enables the definition of a new scenario using a flexible workflow. With this, the original scenario is extended to include the following features:

- Interconnection between the computers where the activity takes place.
- Unbalanced distribution of resources. E.g., not all the computers necessarily have a speech-recognition engine available all the time.
- Possibility for the teacher to change the sequence that is followed during an activity: adding new tasks, modifying tasks, changing certain rules etc.

4.1 Instance for the Integrated Framework for Workflows and Rules

The methodology proposed in this study was followed to develop this scenario. First, the pedagogical rules that govern the system were clearly defined. In parallel to this, the sequence of tasks was also defined. Each rule is specified by an ID, with the aim of categorizing the purpose of each rule and detecting possible dependency between rules. There are two types of rules: rules that must be followed for there to be a specific transition between tasks in the workflow (IR), and rules that must always be followed for there to be any transition between tasks in the workflow (RR). The repetition of a figure in the ID indicates that there is dependency between rules. For example, rule 05IR depends on rule 055IR.

4.2 Instance of the Validation Model

To validate the implementation of the proposed methodology, it is necessary to validate the fulfilment of all the elements that comprise the pedagogical context. In this case, this corresponds to the pedagogical rules and respective sequence of tasks. The evaluation criteria used will correspond to properties that can be evaluated in a Petri-nets model (Table 1).

Table 1. Elements to evaluate and properties evaluated in the respective elements.

Element to be evaluated	**Properties of the element evaluated**
Adherence to pedagogical rules	Reachability, Deadlock-freeness
Adherence to expected sequence	Safety-liveness (using LTL), Deadlock-freeness and Reachability (derived from the adherence to the pedagogical rules)

4.3 Defining Alternative Situations

One aspect to evaluate is the robustness of the proposal presented in this study when faced with the incorporation of new tasks and changes to existing tasks. With this, a set of alternative situations to the existing sequence determined by the workflow and rules is defined. These are then incorporated into the modelling of the problem. When incorporating these situations, it must be verified that there are no inconsistencies with the pre-defined rules. Adherence to the pedagogical rules and expected sequence must also be evaluated, based on the properties included in Table 1. The definition of the alternative situations can be found in Appendix I, with the respective rule that triggers each one. A nomenclature like that used for rules was adopted to identify the alternative situations, using the keyword RP.

4.4 Test Cases

To carry out the validation, simulations are performed using the Petri-nets that were generated based on the transformation methodology proposed in Sect. 3.4. Different models were created based on the definition of the proposed pedagogical scenario: one that considers the original sequence of tasks, and three others that each separately incorporate an alternative situation. With this, the robustness of the modelling can be analyzed separately for each of the proposed alternative situations.

The use of 1, 2, 3, 4 and 5 nodes were considered for evaluating each of these models. A limited number of nodes were used as it measures the robustness of the modelling without considering requirements of scalability. According to the definition of the scenario, each node corresponds to a computer shared by 3 students. Each configuration is tested using both valid and invalid input sets with regards to the specifications of the scenario. The respective models were coded using LTL to evaluate adherence to both the expected execution sequence, as well as the execution sequences deriving from the alternative situations. This code is used by the model-checker to evaluate the validity of the Petri-net in following the tested sequence. Furthermore, the model-checker also generates the entire set of combinations for the order of tasks for each of the test cases.

4.5 Results

As described in Sect. 3, the analysis of the results is focused on the behavior of the Petri-net with regards to the properties that it must fulfil. In the proposed scenario, this analysis is performed for the aspects detailed below. Together, these aspects allow the validity of the modelling to be checked in terms of its translation to Petri-net, as well as in terms of the properties that must be fulfilled when incorporating changes into the original formulation, as described in Table 1. The aspects are as follows:

- Appearance of inaccuracies (i.e., violations of the rules) in the model for all the test cases. This is implicit in the reachability analysis and LTL.
- Detection of deadlock-freeness problems when expected.
- Detection of reachability (i.e., fulfilment of the expected states).
- Detection of problems in LTL (i.e., if an expected sequence is not adhered to).

4.6 Detection of Deadlocks

The appearance and detection of deadlocks was observed when there was no alternative to the use of the speech recognition engine. No other deadlocks appeared (Table 2).

Table 2. Results of the deadlock-freeness analysis.

Scenario	Scenario	Scenario	Scenario	Scenario	Scenario
1 node	1 node	1 node	1 node	1 node	1 node
2 nodes	2 nodes	2 nodes	2 nodes	2 nodes	2 nodes
3 nodes	3 nodes	3 nodes	3 nodes	3 nodes	3 nodes
4 nodes	4 nodes	4 nodes	4 nodes	4 nodes	4 nodes

4.7 Reachability Analysis and LTL

Reachability analysis is performed implicitly when evaluating the expected sequence order using LTL. This is because not adhering to the sequences ensures that not all the expected states are reached in the Petri-nets. In particular, the following situations are observed as causing problems with reachability:

- When the order of the tasks in the original sequence is automatically altered by the model-checker (i.e., models generated by the model-checker vs. expected sequences).
- When there is incorrect input. With this, not all the expected states are reached.

In the reachability analysis for the cases that were used to test the proposed modelling it can be observed that all the expected states are achieved when inserting alternative situations (Table 3).

Table 3. Results of reachability and LTL analysis.

Scenario	1 node	2 nodes	3 nodes	4 nodes	5 nodes
Change in order of tasks	Fail	Fail	Fail	Fail	Fail
Incorrect input	Fail	Fail	Fail	Fail	Fail
Original	Ok	Ok	Ok	Ok	Ok
Original + 10RP	Ok	Ok	Ok	Ok	Ok
Original + 11RP	Ok	Ok	Ok	Ok	Ok
Original + 12RP	Ok	Ok	Ok	Ok	Ok

5 Conclusions

The research question “Is it possible to incorporate complex rules into an imperative workflow so as to allow dynamic changes (during run-time), both to the properties that govern it as well as the order in which the tasks are defined and performed, guaranteeing the integrity of the rules that govern the problem’s domain?”, was answered by creating an integrated framework when developing an imperative workflow. This framework allows rules to be explicitly incorporated, as well as offering great flexibility in a distributed setting. This was achieved by proposing a definition of the workflow using BPEL and integrating rules using service orientation approach. The changes that were introduced can be introduced at any point of the workflow and rules. These changes include adding and eliminating tasks, changing a rule, and changing the order of the tasks, and must be programed in the same language as the language used to program the workflow and rules. Both the workflow and rules are translated to Petri-nets to check that these changes are consistent. This answers the research question “Is it possible to define a methodology for automatic property verification based on Petri-nets for the workflow proposed in the first research question?” To do so, a methodology was proposed in which the workflow and rules are translated to Petri-nets and then integrated into a single network. This allows the validity of the changes that were introduced to be evaluated jointly, verifying intrinsic properties of the Petri-nets and Linear Temporal Logic.

Based on the execution of the example presented in this study, it was seen that the proposal is robust when incorporating alternative situations into the original sequence of tasks. Alternative situations were formulated based on the requirements of the problem, with the observation that the expected results were achieved in terms of the properties of deadlock-freeness, reachability, and safety-liveness. The latter of these is a critical property as it considers the adherence to an expected order of the tasks, meaning that the other properties depend on safety-liveness. In the context of the application presented in this study, the use of collaborative learning allowed for more complex dynamics to be developed, including sequences with interdependency between the activity’s participants. With this, any change to the order of the tasks has a high incidence in the non-fulfilment of safety-liveness. This is contrasted with the incorporation of new tasks within the sequence. In that case, the original sequence is maintained but new possibilities are added to the activity. These can bring with them possible pedagogical requirements, or indeed they can correct certain problems that could have otherwise hindered the fulfilment of the pre-defined requirements.

As shown in Sects. 3 and 4, certain steps of the methodology were done semi-automatically. This made it difficult to use the methodology transparently in a real setting. It remains as future work to build a workflow engine that automatically carries out all the stages in the methodology. Another limitation is related to the reach of the requirements of the problem that was presented in this study, as is a scalability analysis. With this, the growing incorporation of more multimedia resources and a larger number of nodes remains as future work. Finally, another limitation has to do with the generic nature of the scenario that was presented. Although a pedagogical setting was chosen as the basis for this study, the reach of the proposed methodology is not restricted just to this context. However, the chosen scenario must have characteristics, which allow the requirements relating to flexibility to be formed, as well as the definition of the rules that govern the domain of the problem, with the aim of satisfying the model presented in this study.

References

1. Hofmann, M.,Betke, H., Sackmann, S.: Automated Analysis and Adaptation of DRP 266 Long Paper-Decision Support Systems (2004)
2. Macfadyen, L.P., Lockyer, L., Rienties, B.: Learning design and learning analytics: Snapshot 2020. J. Learn. Anal. **7**(3), 6–12 (2020). https://doi.org/10.18608/JLA.2020.73.2
3. Online, R., Agostinho, S., Bennett, S., Lockyer, L., Jones, J., Harper, B.: Learning designs as a stimulus and support for teachers' design practices. http://ro.uow.edu.au/sspapers/422
4. Educational Technology & Society. http://www.ifets.info/
5. Asensio-Pérez, J.I., et al.: Towards teaching as design: exploring the interplay between full-lifecycle learning design tooling and Teacher Professional Development. Comput. Educ. **114**, 92–116 (2017). https://doi.org/10.1016/j.compedu.2017.06.011
6. Celik, D., Magoulas, G.D.: A review, timeline, and categorization of learning design tools. In: Chiu, D.K.W., Marenzi, I., Nanni, U., Spaniol, M., Temperini, M. (eds.) ICWL 2016. LNCS, vol. 10013, pp. 3–13. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-47440-3_1
7. Pozzi, F., Asensio-Perez, J.I., Ceregini, A., Dagnino, F.M., Dimitriadis, Y., Earp, J.: Supporting and representing learning design with digital tools: in between guidance and flexibility. Technol. Pedagog. Educ. **29**(1), 109–128 (2020). https://doi.org/10.1080/1475939X.2020.1714708
8. Persico, D., Pozzi, F.: Informing learning design with learning analytics to improve teacher inquiry. Br. J. Educ. Technol. **46**(2), 230–248 (2015). https://doi.org/10.1111/bjet.12207
9. Torres, J., Cárdenas, C., Dodero, J.M., Juárez, E., Rosson, M.B.: Educational modelling languages and service-oriented learning process engines. Adv. Learn. Process. 17–38 (2010)
10. Koper, R., Miao, Y.: Using the IMS LD standard to describe learning designs. In: Handbook of Research on Learning Design and Learning Objects. IGI Global (2011). https://doi.org/10.4018/9781599048611.ch003
11. Vesin, B., Mangaroska, K., Giannakos, M.: Learning in smart environments: user-centered design and analytics of an adaptive learning system. Smart Learn. Environ. **5**(1), 1–21 (2018). https://doi.org/10.1186/s40561-018-0071-0
12. Magnisalis, I., Demetriadis, S.: Extending IMS-LD capabilities: a review, a proposed framework and implementation cases. Intell. Adapt. Personal. Tech. Comput.-Support. Collab. Learn. 85–108 (2012)

13. Hermans, H., Janssen, J., Koper, R.: Flexible authoring and delivery of online courses using IMS learning design. Interact. Learn. Environ. **24**(6), 1265–1279 (2016). https://doi.org/10.1080/10494820.2014.994220
14. Pérez-Sanagustín, M., Santos, P., Hernández-Leo, D., Blat, J.: 4SPPIces: a case study of factors in a scripted collaborative-learning blended course across spatial locations. Int. J. Comput. Support. Collab. Learn. **7**(3), 443–465 (2012). https://doi.org/10.1007/s11412-011-9139-3
15. Garreta-Domingo, M., Hernández-Leo, D., Sloep, P.B.: Education, technology and design: a much needed interdisciplinary collaboration. In: Kapros, E., Koutsombogera, M. (eds.) Designing for the User Experience in Learning Systems. HIS, pp. 17–39. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-94794-5_2
16. van der Aalst, W.M.P., Pesic, M., Schonenberg, H.: Declarative workflows: balancing between flexibility and support. Comput. Sci. Res. Dev. **23**(2), 99–113 (2009). https://doi.org/10.1007/s00450-009-0057-9
17. Baiyere, A., Salmela, H., Tapanainen, T.: Digital transformation and the new logics of business process management. Eur. J. Inf. Syst. **29**(3), 238–259 (2020). https://doi.org/10.1080/0960085X.2020.1718007
18. la Rosa, M., van der Aalst, W.M.P., Dumas, M., Milani, F.P.: Business process variability modeling: a survey. ACM Comput. Surv. **50**(1) (2017). Association for Computing Machinery. https://doi.org/10.1145/3041957
19. Murguzur, A., Intxausti, K., Urbieta, A., Trujillo, S., Sagardui, G.: Process flexibility in service orchestration: a systematic literature review. Int. J. Coop. Inf. Syst. **23**(3) (2014). https://doi.org/10.1142/S0218843014300010
20. Kloos, C.D., et al.: SmartLet: learning analytics to enhance the design and orchestration in scalable, IoT-enriched, and ubiquitous smart learning environments. In: ACM International Conference Proceeding Series, October 2018, pp. 648–653 (2018). https://doi.org/10.1145/3284179.3284291
21. Andrews, K., Steinau, S., Reichert, M.: Enabling runtime flexibility in data-centric and data-driven process execution engines. Inf. Syst. 101 (2021). https://doi.org/10.1016/j.is.2019.101447
22. Seyffarth, T., Kuehnel, S.: Maintaining business process compliance despite changes: a decision support approach based on process adaptations. J. Decis. Syst **31**(3), 305–335 (2022). https://doi.org/10.1080/12460125.2020.1861920
23. Hasic, F., de Smedt, J., vanden Broucke, S., Serral, E.: Decision as a Service (DaaS): a service-oriented architecture approach for decisions in processes. IEEE Trans. Serv. Comput. **15**(2), 904–917 (2022). https://doi.org/10.1109/TSC.2020.2965516
24. Kittel, K., Sackmann, S., Betke, H., Hofmann, M.: Achieving flexible and compliant processes in disaster management. In: Proceedings of the Annual Hawaii International Conference on System Sciences, pp. 4687–4696 (2013). https://doi.org/10.1109/HICSS.2013.71
25. Aalst, W.M.P.: Everything you always wanted to know about petri nets, but were afraid to ask. In: Hildebrandt, T., van Dongen, B.F., Röglinger, M., Mendling, J. (eds.) BPM 2019. LNCS, vol. 11675, pp. 3–9. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26619-6_1
26. Vidal, J.C., Lama, M., Bugarín, A.: OPENET: ontology-based engine for high-level Petri nets. Expert Syst. Appl. **37**(9), 6493–6509 (2010). https://doi.org/10.1016/j.eswa.2010.02.136
27. Nagl, C., Rosenberg, F., Vitalab, D.: VIDRE-A Distributed Service-Oriented Business Rule Engine based on RuleML. http://www.vitalab.tuwien.ac
28. Wolf, K.: Petri net model checking with LoLA 2. In: Khomenko, V., Roux, O.H. (eds.) PETRI NETS 2018. LNCS, vol. 10877, pp. 351–362. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-91268-4_18

29. Calderón, J.F., Nussbaum, M., Carmach, I., Díaz, J.J., Villalta, M.: A single-display groupware collaborative language laboratory. Interact. Learn. Environ. **24**(4), 758–783 (2016). https://doi.org/10.1080/10494820.2014.917111