Ontology-Based Behavioral Constraint Authoring

Christoph Czepa*, Huy Tran*, Uwe Zdun*, Thanh Tran Thi Kim†, Erhard Weiss† and Christoph Ruhsam†
*Faculty of Computer Science
University of Vienna, Vienna, Austria
Email: {christoph.czepa, huy.tran, uwe.zdun}@univie.ac.at
†Isis Papyrus Europe AG, Maria Enzersdorf, Austria
Email: {thanh.tran, erhard.weiss, christoph.ruhsam}@isis-papyrus.com

***Abstract*—A major approach for formalizing business policies or compliance rules (e.g., stemming from regulatory laws or standards) are behavioral constraints. Flexible business process management approaches such as Adaptive Case Management provide business users the necessary freedom to react to unforeseeable circumstances by ad-hoc changes, but behavioral constraints are often defined and maintained on a technical level which is inaccessible for business users. Consequently, long update cycles of these constraints might result in the enactment of obsolete, incomplete or faulty constraints which hinder the work of the business user instead of supporting it. In this paper, an ontology-based approach for defining and maintaining behavioral constraints in the context of flexible business processes is proposed. The approach aims at enabling business users to take active part in the creation and maintenance of behavioral constraints. The practical applicability of the approach is discussed by means of a realistic scenario on compliance in the context of renovation, repairs, and maintenance of buildings.**

I. Introduction

Many business domains are subject to a large amount of compliance requirements stemming from sources such as regulatory laws (e.g., Sarbanes-Oxley), standards (e.g., ISO 45001 - Occupational health and safety) or best practices (e.g. ITIL). The classical approach for integrating such compliance requirements in the enactment of business processes are flow-driven, predefined business process models that become instantiated and executed. Assuming the business process model is correctly defined and the business users follow it exactly, we can be sure that business process instances are compliant. In reality, however, deviations from those predefined business processes become frequently necessary. This has led to flexible business process management approaches such as *Adaptive Case Management* ([1], [2]) that enable business users to actively shape the enactment of business processes by skipping activities, performing ad-hoc activities, and defining and changing goals. Several researchers propose approaches that we can broadly collected under the umbrella term *Behavioral Constraints* of business processes. Pesic & van der Aalst propose a declarative approach called *Declare* for flexible business process management that offers a graphical specification language to loosely define business processes by relations between activities such as a 'response relation' or 'precedence relation' with underlying formal representations in Linear Temporal Logic (LTL) or Event Calculus ([3], [4]). Ly et al. propose a different graphical language called *Compliance Rule Graphs* with elements such as 'antecedent occurrence' and 'consequence occurrence' [5]. Hildebrandt et al. propose *Dynamic Condition Response Graphs* for trustworthy Adaptive Case Management which is another graphical language comprising relations such as a 'condition relation' and a 'response relation' [6]. Schönig & Zeisig propose *DPIL (Declarative Process Intermediate Language)*, a scripting language which allows the definition of macros such as 'sequence(a, b)' that become enacted by a rule engine.

What all these approaches have in common is a recurring set of *behavioral constraint patterns*, most often with a strong *temporal* focus. Dwyer et al. identified a collection of fundamental temporal patterns such as 'response' and 'precedence' [7]. It is not surprising that such temporal aspects play a major role in business process management, since the order of execution of specific activities due to dependencies between activities is often of importance.

Above-mentioned behavioral constraint approaches aim at being user-friendly by offering graphical notations or textual specification languages that are based on patterns that have their origin in temporal logics (cf. [7], [8], [9]). However, supporting the user by integrating domain knowledge has not yet been considered, which poses an obstacle for end-user acceptance. A more general approach that does not particularly focus on temporal aspects of business processes is the Semantics of Business Vocabulary and Business Rules (SBVR) standard. It is an adopted standard of the Object Management Group (OMG) which enables the use of domain knowledge for the definition of rules. While this standard enables the definition of a vast amount of different rules, the automated verification of them, especially in a non-static, temporal context, is not yet fully solved: The SBVR standard document in the latest version 1.3 (May 2015) states that "[...] *capturing the formal semantics in an appropriate logic (e.g., a temporal or dynamic logic) is a harder task. One possibility is to provide a temporal package that may be imported into a domain model, in order to provide a first-order logic solution. Another possibility is to adopt a temporal modal logic (e.g., treat a possible world as a sequence of accessible states of the fact model). It may well be reasonable to defer decisions on formal semantics for dynamic rules to a later version of the SBVR standard.*" Consequently, there seems to be a link missing between behavioral constraint approaches—which have a strong focus on dynamics of business processes (i.e., temporal relationships)—and ontology-based constraint

approaches—which have a strong focus on static rules of business processes.

In this paper, we propose an *ontology-based behavioral constraint authoring* approach which has the ontology of a domain linked with the application ontology of the business process management solution (which provides the functionality of specific tasks) and allows for using well-known domain concepts in the definition of behavioral constraints. Based on this explicitly defined domain knowledge, business users can define behavioral constraints on basis of the well-known concepts and relations of their business domain, supported by an ontology-fetching auto-completion and suggestion feature. The ontology can be adapted by creating derived concepts from existing ones that are used in existing behavioral constraints. Any such adaptation has a direct effect on the enactment of the constraint, as the derived concept is considered in the enactment of the constraint. By deriving domain-specific concepts from more generic concepts such as an activity or goal, the enactment of business processes becomes interfaced with the ontology. Consequently, business process elements, such as activities, goals and data objects, have in the background an organized structure, the ontology, which can be leveraged to enact ontology-based behavioral constraints while business processes are executed. The core benefits of the approach are (1) the possibility to modify behavioral constraints indirectly by modifying the ontology, which can have an impact on existing rules, (2) the authoring of behavioral constraints in the language of the business domain, and (3) the underlying temporal logic pattern-based approach that enables an automated verification of defined behavioral constraints.

II. Motivating Example

The building and construction industry is subject to a vast amount of compliance rules stemming from sources such as regulatory laws and standards. Let us consider a compliance document published by the EPA (United States Environmental Protection Agency) in 2011 regarding how to cope with potentially lead-contaminated paint during renovations, repairs and maintenance of buildings [10]. The compliance guide applies to all activities that disturb painted surfaces (short DPS) in residential houses, apartments and child-occupied facilities such as schools and day-care centers built before 1978 in the US. Let us further consider a process management application of a company that manages construction, maintenance, repair and renovation work in cooperation with many and changing subcontractors from diverse professions such as plumbers, carpenters and electricians. The company seeks to avoid non-compliance (to maintain its good reputation and avoid litigation) by assisting the subcontractors in staying compliant. Consequently they try to implement EPA's compliance guide by defining several behavioral constraints that become automatically enacted by the process management software.

One problem the company is facing now is determining which tasks, that are potentially performed by the subcontractors, are DPS activities. They recognize that the subcontractors know better which of their activities fall into this category. Since the pool of existing subcontractors is already large and assumed to be growing, and new DPS activities might evolve over time, the management demands a viable solution that reliably captures all future DPS activities in order to ensure compliance with EPA rules.

The approach proposed in this paper enables the company to specify behavioral constraints based on the more general and rather abstract concept of "disturbing painted surfaces", whereas the subcontractors are enabled to relate their concrete, specific DPS activities to this more general and abstract concept. Consequently, the company provides the framework for the subcontractors to support them in achieving compliance with EPA rules, and the subcontractors are encouraged to properly document their domain knowledge in an explicitly defined ontology.

III. Approach

The general idea of ontology-based behavioral constraints is the integration of behavioral constraint enactment in business process management with domain-specific ontologies to directly leverage domain knowledge in behavioral constraints. At the runtime of business processes, case instance elements such as instances of tasks, goals and data are at the same time instances of ontology concepts. Consequently, it is known for every such instance what concept is behind. Based on the structured information of the ontology, users can define ontology-based behavioral constraints, and instances of behavioral constraints can be evaluated by considering state changes over time and the background knowledge given by the ontology.

The approach comprises the following core components:

- The *Ontology* consists of the ontology of the application, which is the interface to the business process enactment engine, and a domain-specific ontology, which refines the application ontology of the business process enactment engine, and can be used to capture additional domain-specific knowledge explicitly.
- *Behavioral Constraints* are derived from compliance rules or best practices and defined on the basis of domain knowledge stemming from the ontology.
- *Constraint Patterns* with sound underlying formal representations are the foundation for the creation of behavioral constraints.
- The *Constraint Editor* is a tool for authoring of behavioral constraints. It is based on constraint patterns and the ontology to support the business user during constraint authoring.
- *Case Enactment* is realized as flexible business process enactment approach (e.g., Adaptive Case Management). There exists a tight connection between case enactment and the ontology of the domain, so that case elements (such as activities, goals and data) are conceptually defined in, and instances stemming from, an ontology.
- *Constraint Enactment* supports the business user by evaluating concrete instances of behavioral constraints during case enactment.

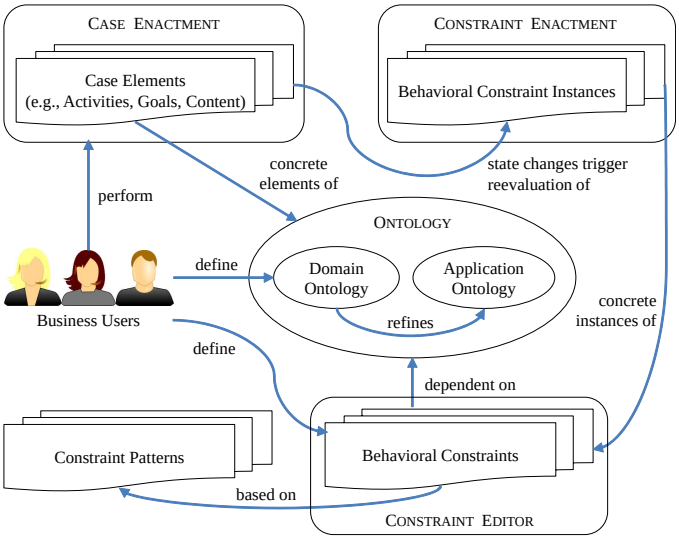

Fig. 1: Approach Overview

- *Business Users* define the knowledge of their domain, handle cases, and are enabled to define behavioral constraints by our approach.

Figure 1 contains an overview on the proposed approach. *Business Users* build up and maintain the business ontology of their business domain (*Domain Ontology*) which refines business process management concepts (defined in the *Application Ontology*) by domain-specific concepts, and they create *Behavioral Constraints* by leveraging the *Domain Ontology* and a set of *Constraint Patterns*. While the domain ontology is completely adaptable, the set of constraint patterns is dependent on underlying formal checking techniques and must be defined by users with appropriate technical background. During *Case Enactment*, business users perform activities to work towards goals. Of course, this involves also the creation and modification of contents (data objects of a case). All these concrete case elements (aka instances) are stemming from ontology concepts. Every time case elements change state, the *Constraint Enactment* component reevaluates the instances of behavioral constraints. Since case elements are instances of domain ontology concepts, behavioral constraint instances can be evaluated by querying domain knowledge (taxonomy, relationships) from the ontology.

The creation of ontology-based behavioral constraints by business users using the constraint editor is shown schematically by a sequence diagram in Figure 2. The constraint editor makes extensive use of the ontology while the business user types a constraint. In the beginning, the business user starts typing and the constraint editor requests ontology concepts matching the (incomplete) user inputs (e.g., by matching the starting string of concepts or by computing the Levenshtein distance). The response is a set of concepts which are proposed to the business user. The user either selects a proposed concept or continues typing. Eventually a proposed concept must be selected. Once the business user has selected a concept, it

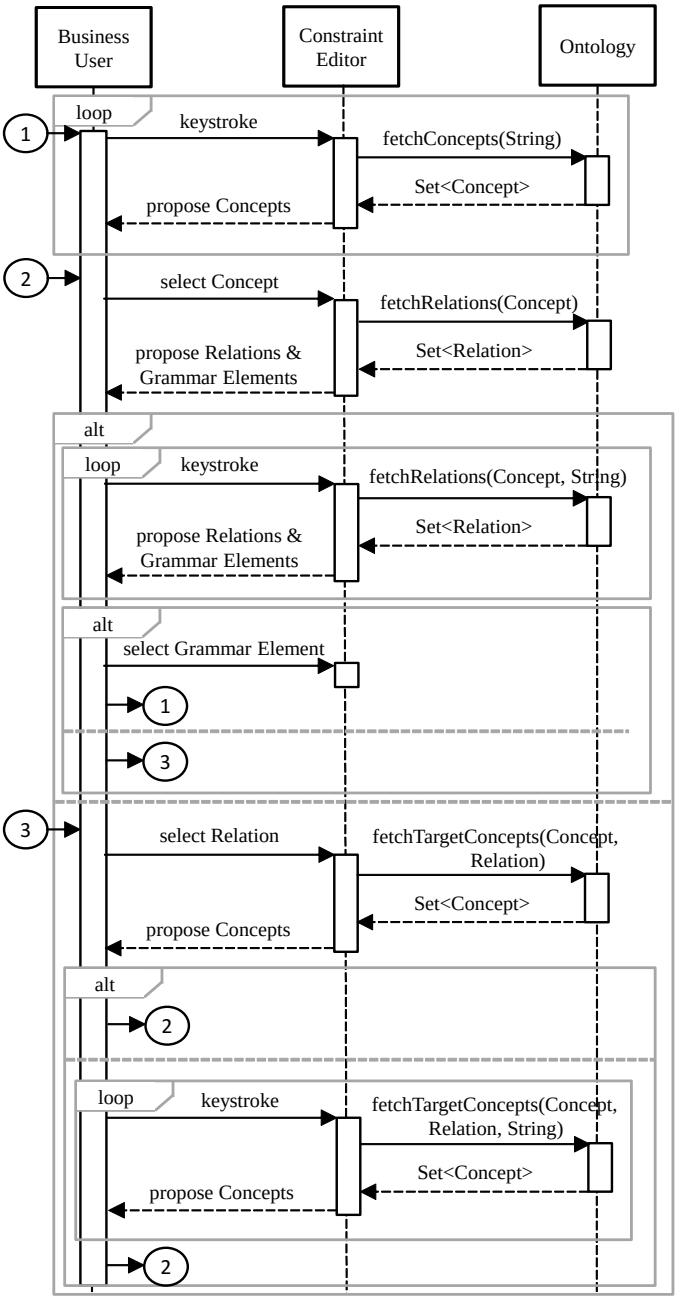

Fig. 2: Sequence diagram for constraint authoring

becomes possible to use the *context* of the specific domain concept. Thus, further auto-completions can be based on the set of *relations* of the concept. If appropriate, the constraint editor makes not only proposals based on the ontology, but also proposes the possible elements of the constraint grammar to the user. When the user selects a specific relation, the context for suggesting further inputs is narrowed down to concepts that are potential targets of this relation originating from the already selected concept. Figure 2 continues with processing user inputs to update the set of possible target concepts in the current relation context. Once the user has selected a target concept, the next part of the constraint

could be either an element of the constraint grammar or a relation. For concepts that are derived from the application concepts *Goal* and *Activity*, the specific *runtime state* of these concepts can be directly specified after the name of each concept. Standard runtime states of goals are *initiated* and *completed*, and standard runtime states of activities are *started* and *finished*.

Constraint patterns stemming from different sources (such as Dwyer et al. [7], Elgammal et al. [11] and van der Aalst & Pesic [12]) can be integrated. For example, *Response* describes a temporal cause-effect constraint which can be expressed as

```
Expression I requires Expression II later
```

where *Expression I* is the cause and *Expression II* is the effect. An *Expression* is defined based on the ontology and allows leveraging *domain-specific knowledge* for the creation of behavioral constraints. The following ontology elements can be used to define such an ontology-based expression:

- *Concepts*: Concepts are the anchor for defining ontology expressions, so every ontology-based expression must start with a concept. If a concept is derived from a *Goal* or *Activity*, then the *runtime state* can be specified directly after the name of the concept.
- *Relations*: Once the context of a concept is defined, the relations of this concept to other concepts become obvious and usable.
- *Constraint Concepts*: Constraint concepts are specialized child concepts that allow defining specific constraints that determine whether an instance of the parent concept is also an instance of the constraint concept. These constraints may be related to attributes and/or relations of the parent concept.

During case enactment, ontology expressions are evaluated based on the actual instances and relations that are present in a case instance. An ontology expression consisting of a (constraint) concept $\mathcal{C}$ is matched by $\mathcal{E}$ iff $\exists i \mid i \in \mathcal{I} \wedge i : \mathcal{C}$ where $\mathcal{I}$ is the set of case elements of a case instance $\mathcal{E}$ that are instances of ontology elements, and $\mathcal{I}$ comprises states of activities and goals as well as states of data objects by representing a snapshot of the case instance at the instant in time in which an event occurs (i.e., state change of activity, goal or data). An ontology expression consisting of a (constraint) concept $\mathcal{C}$ with a runtime state $\mathcal{C}.s$ is matched by $\mathcal{E}$ iff $\exists i \mid i \in \mathcal{I} \wedge i : \mathcal{C} \wedge i.s = C.s$. An ontology expression consisting of a relation $\mathcal{R}$ is matched by $\mathcal{E}$ iff $\exists (i_1, i_2) : \mathcal{R} \mid i_1 \in \mathcal{I} \wedge i_2 \in \mathcal{I}$.

IV. Practical Scenario

This section extends the motivating example (cf. Section II) and discusses how the proposed approach can be applied to realize automated support for enacting EPA rules [10]. Figure 3 contains an ontology that covers both the abstract concept of "disturbing painted surfaces" (DPS) and specific DPS activities of subcontractors. The concept *Disturbing Painted Surfaces* plays the special role of a *constraint concept* and represents some *On Site Activity* that disturbs *Painted Surfaces*. Consequently all concepts derived from *Disturbing Painted Surface* have implicitly a relation *disturbs* to the concept *Painted Surfaces*. Moreover, concepts that have a relation *disturbs* to *Painted Surface* and are at the same time derived from an *On Site Activity* concept (that includes children of the *Activity* concept) represent at the same time a *Disturbing Painted Surfaces* concept. Child activities like *Grind Wall* can be derived from this concept. Alternatively, the concept *Pry Open Wall* has the relation *disturbs* and is derived from the more general concept *On Site Activity*. By having the *disturbs* relation and by being at the same time an *On Site Activity*, it is also a *Disturbing Painted Surfaces* concept.

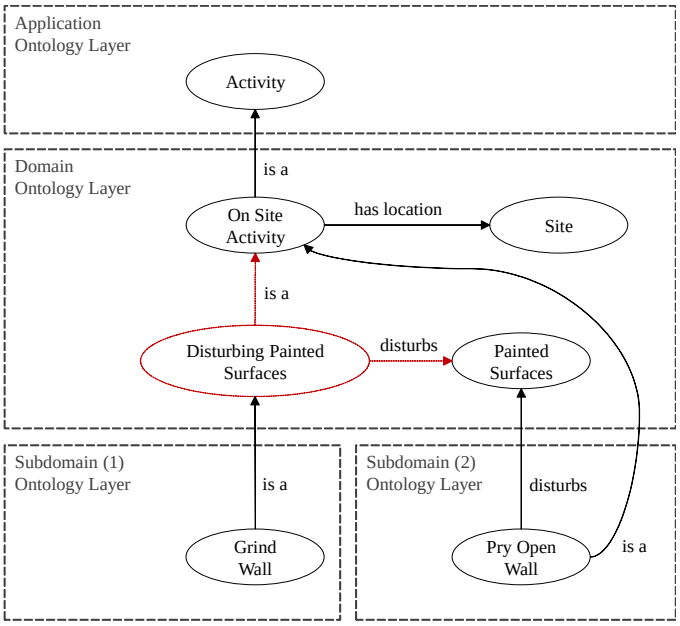

Fig. 3: Layered ontology for "disturbing painted surfaces" activities

The sample ontology in Figure 3 is organized in layers. Subcontractors are enabled to create and maintain their own *Subdomain Ontology Layer* that builds upon the *Domain Ontology Layer*. This layer is maintained by the umbrella organization that employs the subcontractors. The *Application Ontology Layer* is the interface to the business process management software. By having the freedom to modify the ontology of the subdomain, subcontractors can add new activities to the ontology any time and benefit from the behavioral constraint support that has been defined on the basis of existing concepts.

For that reason ontology-based behavioral constraints are derived from the EPA compliance guide. EPA rules are only applicable to "*activities that disturb painted surfaces in a home or child-occupied facility built before 1978*". Consequently it must be checked whether the building is a home or child-occupied facility built before 1978 prior to disturbing its surfaces, so as to possibly take precautions due to potential lead contamination. A business user can create such behavioral constraints by using the already existing, more general domain

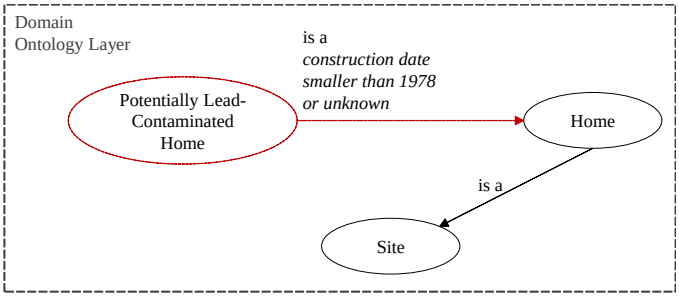

Fig. 4: Constraint concept with constraint defined on attribute of the parent concept

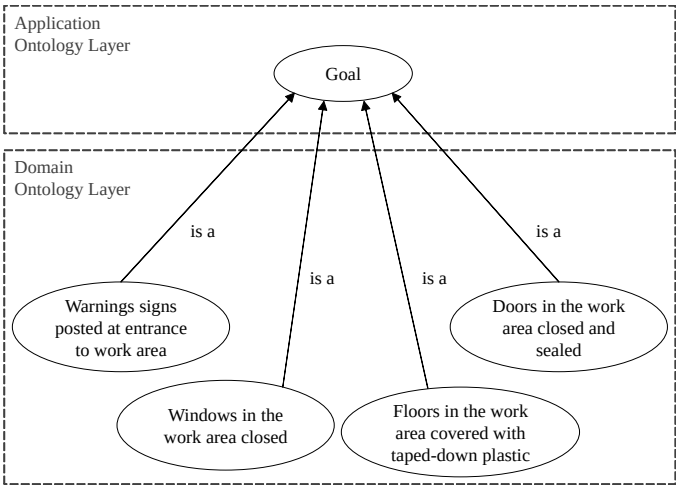

Fig. 5: Creating specific goals for EPA rules

concepts *House* and *Child-Occupied Facility*, and derive the new constraint concepts *Potentially Lead-Contaminated Home* and *Potentially Lead-Contaminated Child-Occupied Facility* that become effective whenever the construction date attribute of the parent concept is smaller than 1978 or not yet defined (Figure 4).

Flexible business process approaches like ACM are *goal-driven*. Consequently several goals related to preparing the work area must be reached in advance of disturbing potentially lead-contaminated surfaces. These goals are (list contains examples and is not meant to be exhaustive):

- Warning signs posted at entrance to work area
- Floors in the work area covered with taped-down plastic
- Doors in the work area closed and sealed

Business users can create such goals by deriving child concepts from the more general *Goal* concept (Figure 5). To make sure that these goals are reached before any potentially lead-contaminated surface is disturbed, the business user makes use of the constraint editor to create behavioral constraints. Listing 1 contains four *Precedence* constraints. Important goals related to the preparations for lead-contaminated work areas must be reached first and only then is it allowed to perform DPS activities. As can be seen for these sample constraints, the controlled natural language grammar of the behavioral constraint language ("operand1 requires operand2 earlier") becomes intertwined with expressions originating from the domain ontology. The operator "has location" stems from the relation with the same designation between the concepts *On Site Activity* and *Site*. The behavioral constraints are close to natural language, which aims at making them comprehensible for business users, but they still follow a structured, well-defined scheme comprising the constraint grammar and the ontology, which is the basis for automated tool support for constraint enactment.

```
Disturbing Painted Surfaces started has location
    Potentially Lead-Contaminated Home or Potentially
    Lead-Contaminated Child-Occupied Facility requires
    Warning signs posted at entrance to work completed
    earlier

Disturbing Painted Surfaces started has location
    Potentially Lead-Contaminated Home or Potentially
    Lead-Contaminated Child-Occupied Facility requires
    Floors in the work area covered with taped-down
    plastic completed earlier

Disturbing Painted Surfaces started has location
    Potentially Lead-Contaminated Home or Potentially
    Lead-Contaminated Child-Occupied Facility requires
    Doors in the work area closed and sealed completed
    earlier
```

Listing 1: Behavioral constraints related to preparations

Automated support for enacting behavioral constraints can be provided by translating the constraints to a formal verification language. Possible formalisms and techniques include Computation Tree Logic (CTL), LTL (Linear Temporal Logic), EPL (Event Processing Language) and EC (Event Calculus). For the proposed ontology-based behavioral constraint approach, we do not intend to prescribe a specific underlying formalism or verification technique. Nevertheless, we would like to showcase how the enactment of ontology-based behavioral constraints works from beginning to end. For this purpose we use LTL, since it is an established formal language for the definition of desired properties of hardware and software systems [13]. LTL has become a *de facto* standard in business process verification due to the possibility of translating LTL formulas to nondeterminstic finite automata (NFAs) for *runtime verification* ([14], [3]) of business processes, and the extensive use of LTL as a specification language in *model checking* ([15], [16]) of business processes and for *compliance* and *security* modeling ([11], [17]). LTL representations of the behavioral constraints are automatically generated by applying the mappings proposed by Dwyer et al. [7] and by properly resolving propositional variables based on the domain ontology. We will now discuss an exemplary enactment of the first ontology-based behavioral constraint in Listing 1, which is defined as

```
Disturbing Painted Surfaces started has location
    Potentially Lead-Contaminated Home or Potentially
    Lead-Contaminated Child-Occupied Facility requires
    Warning signs posted at entrance to work completed
    earlier.
```

For automatic enactment as LTL specification, the operands

```
Disturbing Painted Surfaces started has location
    Potentially Lead-Contaminated Home or Potentially
    Lead-Contaminated Child-Occupied Facility
```

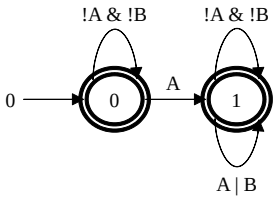

Fig. 6: NFA for *precedence* constraint

denoted as B, and

`Warning signs posted at entrance to work completed`

denoted as A, must be resolved to propositional variables to be able to enact the *Precedence* constraint [7] as LTL formula $\neg B\ \mathcal{W}\ A \equiv \Box\neg B \vee \neg B\ \mathcal{U}\ A$ by an NFA (Figure 6) that represents the formula [14].

As a concrete case let us consider the renovation of a flat in a building which was constructed in 1971. A new case is opened in the business process software for this renovation. The renovator adds *Home* as the site of the renovation and enters the *construction date* 1971. From this moment on, the case instance has an instance of the concept *Home* which is at the same time an instance of *Potentially Lead-Contaminated Home* because of the construction date, which is smaller than 1978. During the enactment of this case, the renovator can perform various ad-hoc activities. For example, when the renovator intends to start wall grinding (by instantiating the *Grind Wall* concept and starting the activity) on this site (*has location* relation), the propositional variable B becomes true. Since the *Grind Wall* instance has the *is a* relation to *Disturbing Painted Surfaces* and the *has location* relation to the *Home* instance, which is also a *Potentially Lead-Contaminated Home* because of the construction date, B is resolved to *true*. Consequently, $B = true$ is the input to the automaton which is in state 0, but the automaton does not accept such an input. Thus, the *Grind Wall* activity would violate the behavioral constraint. Once the renovator enters that warning signs are posted at entrance to work, $A = true$ is sent to the NFA. This causes a state transition of the automaton from state 0 to state 1. From that moment on *Disturbing Painted Surfaces* activities can no longer violate the behavioral constraint since all future inputs are accepted by this automaton.

V. Discussion

The proposed ontology-based behavioral constraint approach enables the use of domain knowledge in behavioral constraints of business processes. As a result of this integration, the behavioral constraints can be formulated in a structured natural language that offers both the elements stemming from behavioral constraint patterns and the elements of the business domain. Since the resulting ontology-based behavioral constraints are formally defined on the basis of the ontology and constraint patterns (with one or more underlying formal verification techniques), automated verification support for the enactment of these constraints becomes possible during the handling of cases by business users. Since the general idea of ontology-based behavioral constraints can be used independently from a specific grammar and notation of behavioral constraints, it might be integrable with existing approaches such as Declare [3] and DPIL [18]. Declare provides a graphical front-end for the definition of behavioral constraints which become directly enacted as declarative workflows (by automata that represent Linear Temporal Logic specifications or by Event Calculus run as Prolog programs). By using the graphical notation of different arrow connectors, the user can define behavioral constraints among activities which are symbolized by boxes with labels. These approaches can integrate the proposed approach by enabling the user to specify ontology expressions instead of those labels. DPIL (Declarative Process Intermediate Language) is a textual language for the specification of behavioral constraints that become enacted by the JBoss Drools rule engine. Instead of defining business process elements such as data objects and activities in DPIL scripts, a user could select such elements from the business ontology. Integrating domain ontologies into those existing behavioral constraint approaches might improve usability for business users.

Opening behavioral constraint authoring and change to business users might be a controversial topic since the actions of business users are at the same time subject to these behavioral constraints. Classically business processes are defined as flow charts (e.g., BPMN models) and business users are ought to follow exactly the steps as prescribed by the process model. This enables a strong control but tempts the business user to perform undocumented actions (i.e., actions that are not disclosed to the IT system) if it becomes necessary to leave the predefined flow of actions. Knowledge-intensive business domains require more flexibility which has led to constraint-based business processes. Those do not longer require the business user to stick to predefined, step-by-step business processes. The main objective of constraint-based business processes is supporting the business user to stay compliant with necessary rules (e.g., regulatory laws, standards) while enabling largest possible flexibility. Thus, behavioral constraints must be seen as a support feature for business users to stay compliant. Behavioral constraints are not meant to hinder the work of the business user. If business users were excluded from having control over behavioral constraints, it would become likely that they again perform undocumented actions just to meet the rules of the IT system.

Ontology-based behavioral constraints create new challenges which are not covered by the work presented in this paper. It is an open research challenge how to keep the behavioral constraints consistent with an evolving ontology. Concepts of the ontology and relations among them might change over time which can lead to disappearance of ontology elements that are part of behavioral constraints. This might have different possible implications for the behavioral constraint: The existence of the behavioral constraint might no longer be necessary, so it may be deleted. Alternatively, there might be still the need for this behavioral constraint but

an adaption to the current ontology is required. If and how business users can be supported to make such an adaption (e.g., by keeping track of ontology changes and leveraging change histories of ontology elements of behavioral constraints) must be further investigated. In general, the maintainability of behavioral constraints by business users must scale even with a growing number of behavioral constraints. These challenges lead to many new interesting possibilities for future research.

Further studies are required on the understandability of the behavioral constraint patterns by business users. It could be of interest to find out whether the total count of possible behavioral constraint patterns offered has an influence on understandability. Of interest is also the influence of a specific grammar of such constraint patterns on the understandability.

VI. Related Work

Yu et al. propose a verification approach for service compositions that are described by BPEL models [19]. Their approach implements a constraint grammar, based on the constraint patterns by Dwyer et al. [7], directly in an ontology. The focus of their work seems to be the realization of these constraint patterns by an ontology, but the connection to domain-specific ontologies is not discussed. Since the set of constraint patterns is maintained by technical users, in our case we do not see the requirement for representing the grammar as part of an ontology. Our approach links the constraint pattern-based grammar with expressions stemming from a domain ontology.

While the previously mentioned approach is pattern-based, an approach by Yan et al. seeks to directly abstract LTL formulas by natural language [20]. In particular, expressions in structured natural language are directly transformed to LTL. However, the controlled grammar of the language still contains LTL operators (globally, always, eventually) that might be hard to comprehend for non-technical business users as language elements. An integration of domain knowledge by leveraging ontologies is not part of their approach.

SBVR (Semantics of Business Vocabulary and Business Rules) is an adopted standard released and maintained by the OMG (Object Management Group) [21]. The intention behind SBVR is to offer a structured natural language approach for the authoring of business rules. However, automated verification support for SBVR is challenging, especially for non-static rules with temporal aspects. Elgammal & Butler propose a manual translation of SBVR rules to an LTL-based graphical compliance rule language [22]. Solomakhin et al. propose a formalization of SBVR rules by first-order deontic-alethic logic (FODAL) to support automated reasoning based on an ontology. Manaf et al. [23] propose SBVR as a specification language for service choreographies. They show with an example that the same rule can be expressed by DecSerFlow patterns [12] (LTL pattern-based specification language for service flows) and SBVR. The DecSerFlow language is also the foundation for the declarative workflow approach Declare [3]. Declarative workflow approaches predominantly enable the definition of behavioral constraints with a temporal context which specify business processes in a declarative manner, but they do not focus on the integration of domain ontologies. Our approach aims at bridging these worlds by enabling business users to define behavioral constraints that have a temporal focus, in an editor that makes use of the domain ontology to allow for defining rules in the well-known business terms of the specific business domain.

The work presented in this paper is also related to the huge body of compliance research. Van der Werf et al. propose "Context-Aware Compliance Checking" [24] by deriving ontologies from audit logs and checking rules in Semantic Web Rule Language (SWRL) [25]. Yip et al. propose a similar approach [26]. Despite using ontologies, their approaches are different from ours since they do not focus on temporal behavioral constraints and constraint authoring in structured natural language. Elgammal et al. propose a Compliance Request Language (CRL) that is formally grounded on temporal logic and mapped to LTL formulas [27]. They identify a set of compliance patterns comprising atomic patterns (extension of the order and occurrence patterns proposed by Dwyer et al. [7]), composite patterns (e.g., mutual exclusion and coexistence), resource patterns (i.e., rules related to roles) and timed patterns (i.e., rules related to quantitative time). CRL does not yet consider ontologies. Awad et al. propose a graphical language called BPMN-Q for the specification of compliance requirements which can be transformed to temporal logic formulas [28], but ontologies are not considered. Ly et al. propose a Compliance Monitoring Functionality Framework (CMFF) which contains ten criteria, and they analyze existing compliance monitoring approaches based on their framework [29]. The framework considers many important compliance aspects such as activity lifecycles, nonatomic activities, data, roles, reactive and proactive support, but it does not specifically focus on specification languages or ontologies.

In summary, it can be said that on the one hand there exists a large body of work on compliance of business processes which does not explicitly consider the integration of ontologies, and on the other hand there are efforts to create business rules on basis of an ontology. Our work is positioned in between. It is grounded on well-established temporal logic patterns that allow an automated verification (like existing compliance and declarative workflow approaches [8], [3]), and it allows to define parts of constraints on basis of knowledge stemming from an ontology (as it is also the goal of standardization efforts in SBVR [21]).

VII. Implementation

The approach is implemented as a prototype extension of the Papyrus Platform[1]. Figure 7 shows a screenshot of the ontology editor and the constraint editor.

VIII. Conclusion and Future Work

The presented approach aims at offering a constraint authoring that is approachable to non-technical users by making use

[1] http://www.isis-papyrus.com

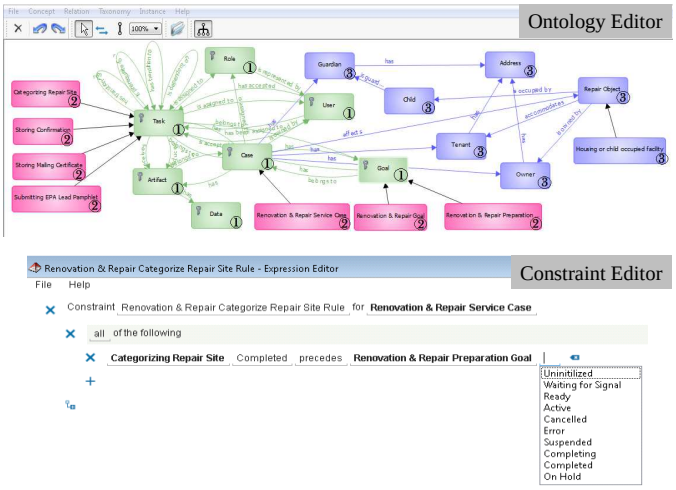

Fig. 7: Ontology editor and constraint editor in Isis Papyrus

of ontology-based domain knowledge. The approach is applied in the context of a realistic scenario (Section IV), but further evaluations are necessary. There exist many assumptions that must be further evaluated: By allowing the business user to actively shape the behavioral constraints, the approach might allow a faster integration and adaption of behavioral constraints, eliminating long delays caused by static maintenance cycles for integrating and modifying behavioral constraints. Consequently, it might be as well avoided that obsolete, faulty and unnecessary constraints are enacted that might drive business users into cheating the process management software by performing undisclosed actions. Thus, the approach might also contribute to a more complete documentation (i.e., audit trails). Moreover, it may encourage business users to capture their tacit knowledge as (non-mandatory) behavioral constraints. The evaluation of those assumptions by qualitative and quantitative studies are opportunities for future research. Moreover, open challenges regarding consistency and maintainability (as outlined in Section V) must be resolved.

Acknowledgment

The research leading to these results has received funding from the FFG project CACAO, no. 843461 and the Wiener Wissenschafts-, Forschungs- und Technologiefonds (WWTF), Grant No. ICT12-001.

References

[1] K. D. Swenson, *Mastering the unpredictable: how adaptive case management will revolutionize the way that knowledge workers get things done*. Meghan-Kiffer Press, 2010.

[2] M. J. Pucher, "Considerations for implementing adaptive case management," in *Taming the Unpredictable Real World Adaptive Case Management: Case Studies and Practical Guidance*, L. Fischer, Ed. Future Strategies Inc., 2011.

[3] M. Pesic and W. M. P. van der Aalst, "A declarative approach for flexible business processes management," in *BPM Workshops*. Springer, 2006, pp. 169–180.

[4] M. Montali, F. M. Maggi, F. Chesani, P. Mello, and W. M. P. v. d. Aalst, "Monitoring business constraints with the event calculus," *ACM Trans. Intell. Syst. Technol.*, vol. 5, no. 1, pp. 17:1–17:30, Jan. 2014.

[5] L. T. Ly, S. Rinderle-Ma, and P. Dadam, *CAiSE, Hammamet, Tunisia*. Springer, 2010, ch. Design and Verification of Instantiable Compliance Rule Graphs in Process-Aware Information Systems, pp. 9–23.

[6] T. Hildebrandt, M. Marquard, R. R. Mukkamala, and T. Slaats, *OTM Workshops, Graz, Austria*. Springer, 2013, ch. Dynamic Condition Response Graphs for Trustworthy Adaptive Case Management, pp. 166–171.

[7] M. B. Dwyer, G. S. Avrunin, and J. C. Corbett, "Patterns in property specifications for finite-state verification," in *ICSE*. ACM, 1999, pp. 411–420.

[8] A. Elgammal, O. Turetken, W.-J. van den Heuvel, and M. Papazoglou, "Formalizing and appling compliance patterns for business process compliance," *Software & Systems Modeling*, vol. 15, no. 1, pp. 119–146, 2016.

[9] W. M. P. van der Aalst and M. Pesic, *DecSerFlow: Towards a Truly Declarative Service Flow Language*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 1–23.

[10] EPA, "Small Entity Compliance Guide to Renovate Right," http://epa.gov/sites/production/files/documents/sbcomplianceguide.pdf, last accessed: July 1, 2016.

[11] A. Elgammal, O. Turetken, and W.-J. Van Den Heuvel, "Using patterns for the analysis and resolution of compliance violations," *International Journal of Cooperative Information Systems*, vol. 21, no. 01, pp. 31–54, 2012.

[12] W. M. P. Aalst and M. Pesic, *WS-FM, Vienna, Austria*. Springer, 2006, ch. DecSerFlow: Towards a Truly Declarative Service Flow Language, pp. 1–23.

[13] A. Pnueli, "The temporal logic of programs," in *Foundations of Computer Science*, Oct 1977, pp. 46–57.

[14] G. De Giacomo, R. De Masellis, and M. Montali, "Reasoning on ltl on finite traces: Insensitivity to infiniteness," in *AAAI*. AAAI Press, 2014, pp. 1027–1033.

[15] R. Eshuis, "Symbolic model checking of uml activity diagrams," *ACM Trans. Softw. Eng. Methodol.*, vol. 15, no. 1, pp. 1–38, Jan. 2006.

[16] Z. Sbai, A. Missaoui, K. Barkaoui, and R. Ben Ayed, "On the verification of business processes by model checking techniques," in *ICSTE*, vol. 1, Oct 2010, pp. V1–97–V1–103.

[17] A. Armando and S. E. Ponta, "Model checking of security-sensitive business processes," in *FAST*. Springer, 2010, pp. 66–80.

[18] M. Zeising, S. Schonig, and S. Jablonski, "Towards a common platform for the support of routine and agile business processes," in *CollaborateCom*, Oct 2014, pp. 94–103.

[19] J. Yu, T. P. Manh, J. Han, Y. Jin, Y. Han, and J. Wang, *WISE 2006, Wuhan, China*. Springer, 2006, ch. Pattern Based Property Specification and Verification for Service Composition, pp. 156–168.

[20] R. Yan, C.-H. Cheng, and Y. Chai, "Formal consistency checking over specifications in natural languages," in *DATE*, 2015, pp. 1677–1682.

[21] OMG, "Semantics of Business Vocabulary and Rules (SBVR)," http://www.omg.org/spec/SBVR/, last accessed: July 1, 2016.

[22] A. Elgammal and T. Butler, *ICSOC 2014 Workshops*. Springer, 2015, ch. Towards a Framework for Semantically-Enabled Compliance Management in Financial Services, pp. 171–184.

[23] N. A. Manaf, S. Moschoyiannis, and P. J. Krause, "Service choreography, sbvr, and time," in *FOCLASA, Madrid, Spain*, 2015, pp. 63–77.

[24] J. M. E. M. Werf, H. M. W. Verbeek, and W. M. P. Aalst, *BPM, Tallinn, Estonia*. Springer, 2012, ch. Context-Aware Compliance Checking, pp. 98–113.

[25] W3C, "SWRL: A Semantic Web Rule Language Combining OWL and RuleML," https://www.w3.org/Submission/SWRL/, last accessed: July 1, 2016.

[26] F. Yip, N. Parameswaran, and P. Ray, "Rules and ontology in compliance management," in *EDOC*, Oct 2007, pp. 435–435.

[27] A. Elgammal, O. Turetken, W.-J. Heuvel, and M. Papazoglou, "Formalizing and applying compliance patterns for business process compliance," *Software & Systems Modeling*, vol. 15, no. 1, pp. 119–146, 2014.

[28] A. Awad, G. Decker, and M. Weske, *BPM, Milan, Italy*. Springer, 2008, ch. Efficient Compliance Checking Using BPMN-Q and Temporal Logic, pp. 326–341.

[29] L. T. Ly, F. M. Maggi, M. Montali, S. Rinderle-Ma, and W. M. van der Aalst, "Compliance monitoring in business processes: Functionalities, application, and tool-support," *Information Systems*, vol. 54, pp. 209 – 234, 2015.