Towards a Compliance Support Framework for Adaptive Case Management

Christoph Czepa*, Huy Tran*, Uwe Zdun*, Thanh Tran Thi Kim†, Erhard Weiss† and Christoph Ruhsam†
*University of Vienna, Faculty of Computer Science, Software Architecture Research Group,
Währingerstraße 29, 1090 Vienna, Austria
Email: {christoph.czepa, huy.tran, uwe.zdun}@univie.ac.at
†Isis Papyrus Europe AG, Alter Wienerweg 12, 2344 Maria Enzersdorf, Austria
Email: {thanh.tran, erhard.weiss, christoph.ruhsam}@isis-papyrus.com

***Abstract*—Current Adaptive Case Management (ACM) solutions are strong in flexibility, but business users must still meet compliance rules stemming from sources such as laws (e.g., Sarbanes-Oxley Act), standards (e.g., ISO 45001) and best practices (e.g., ITIL). This paper presents a framework on how to enable support for compliance in the context of ACM by constraints. Since ACM applications undergo constant change, there must be ways to introduce compliance rules on the fly. Currently, constraints (and similar alternative solutions) are predominately maintained by technical users, which results in long maintenance cycles. Our framework aims at enabling faster adoption of changing compliance requirements, both explicitly by enabling non-technical users (knowledge workers) to define and adapt constraints, and implicitly by learning from the decisions taken by other knowledge workers during case enactments. The former is achieved by supporting domain knowledge, maintained in an ontology. The latter is supported by a recommendation approach that enables an automated knowledge transfer between knowledge workers by propagating tacit knowledge, best practices, and the handling of constraints and their violations.**

I. Introduction & Motivation

Knowledge-intensive processes are challenging: On the one hand, knowledge workers must cope with rapidly changing environments and handle very specific situations which can not be fully designed before the process is executed and require runtime flexibility. On the other hand, the domains in which knowledge work is necessary are often heavily regulated (e.g., the insurance or health sector) and staying compliant is difficult if the IT system does not provide the necessary support to follow the imposed compliance requirements.

Adaptive Case Management (ACM) is a paradigm for flexible, goal- and knowledge-driven business process management [1]. ACM enables knowledge workers to actively shape the way case instances are executed while documenting their actions and providing the needed IT support. ACM solutions must support knowledge workers to stay on the right track instead of hindering their work. Long maintenance cycles might lead to the enactment of outdated compliance rules and new, useful compliance rules might be introduced with month-long delays.

This paper presents a framework that aims at enabling faster adoption of changing compliance requirements, both explicitly by enabling non-technical users (i.e., the so-called knowledge workers) to define and adapt constraints, and implicitly by learning from the decisions made by other knowledge workers during case enactments. The former is achieved by supporting domain knowledge, realized as an ontology which links domain-specific concepts (defined by knowledge workers) to technical concepts of the ACM ontology (defined by technical users). The latter is supported by a recommendation approach that enables an automated knowledge transfer between knowledge workers by propagating tacit knowledge, best practices, and the handling of constraints and their violations. This recommendation approach is a machine learning problem that can be described as a decision learning (classification) problem. The framework describes the steps that are necessary to prepare data from past case enactments for the learning, namely the selection which past case enactments must be considered for achieving a specific purpose (e.g., to overcome a specific constraint violation), and the preprocessing that uses ontology knowledge to prepare information for the learning process, such as temporal relationships (e.g., relative time between the completion of one task and another).

II. Framework Overview

Figure 1 shows an overview of the proposed constraint framework for ACM. The framework comprises one central concept, the ontology, one targeted type of stakeholder, the knowledge worker, and five supported functionalities:

(Central Concept) The *Ontology* comprises the ontology of the ACM domain and the ontology of a business domain (or the ontologies of several business domains). Business domain-specific concepts (e.g., a domain-specific activity) are put in relation to ACM concepts (e.g., a generic user task that is supported by the ACM application). While the ACM ontology is maintained by technical users (e.g., the vendor), knowledge workers can adapt the ontology of their domain. Constraints can be formulated on the basis of the ontology of the domain, which means that knowledge from the domain supports the creation of constraints. During constraint enactment, it is known which concept is a generalization of which other concept and how concepts are related. Consequently, constraints can be evaluated based on existing case elements, such as tasks, goals, and data objects, which are concrete instances of those concepts. For example, instead of recommending very specialized next actions to knowledge workers during case

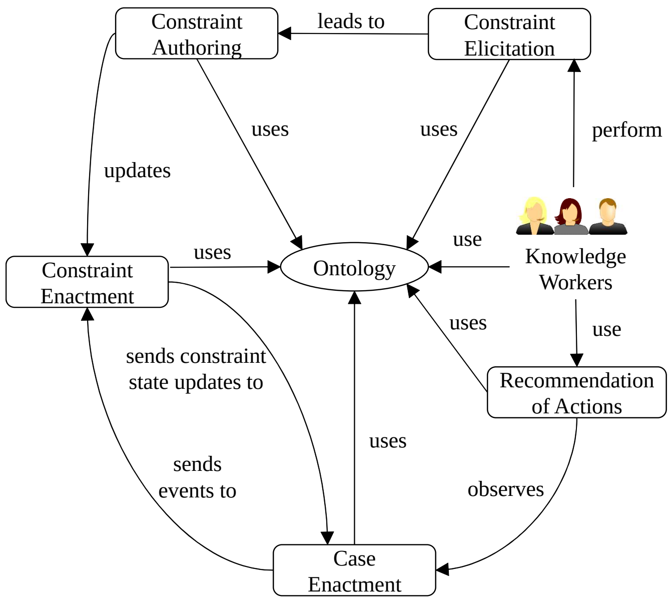


Fig. 1. Overview of Compliance Support Framework

enactment, it probably makes sense to propose a more general approach (i.e., a parent concept) that fosters a greater scope of possibilities instead (e.g., instead of proposing the action 'payment by credit card', the more general action 'payment' can be proposed).

(Targeted Stakeholder) *Knowledge workers* are the main drivers of the ACM system. They handle cases to the best of their knowledge while having to cope with unforeseeable situations commonly. Their knowledge is the foundation of this constraint framework. On the one hand, their knowledge is directly used to define the domain ontology. On the other hand, their knowledge is automatically processed to propagate it to other knowledge workers which enables an automatic knowledge transfer among knowledge workers.

(Functionality 1) *Constraint Elicitation* is concerned with transforming regulatory laws, standards, best practices and tacit knowledge into internal policies, and in a next step to constraints that can be automatically enacted.

(Functionality 2) *Constraint Authoring* is concerned with defining, modifying and managing constraints. The language used to create constraint expression is partly derived from the ontology and can make use of the domain concepts and their relations. Other parts of the constraint language are related to specific constraint patterns (e.g., [2]).

(Functionality 3) *Constraint Enactment* monitors constraint instances. Reacting to constraint modifications and carrying out the necessary steps (e.g., removal of an obsolete constraint instance) are important responsibilities of this component. Constraint enactment reevaluates the states of constraint instances with every new relevant received event. State changes, in particular those that violate constraints, are reported to the knowledge worker.

(Functionality 4) *Case Enactment* comprises the daily business of knowledge workers, namely the handling of cases to progress towards specific goals while coping with unforeseeable circumstances on a regular basis. Documents and data objects are continuously created, modified and reviewed. Tasks can be delegated to and carried out by knowledge workers belonging to different professions. During case enactment, knowledge workers are continuously informed about constraint violations.

(Functionality 5) *Recommendation of Actions* enables automated knowledge transfers between knowledge workers. Whenever the knowledge worker seeks advice from this component, the current circumstances and the history of the case are used to find one or more appropriate candidate actions that are in line with the decisions made by knowledge workers that have been in a similar situation. This component propagates information, such as tacit knowledge, evolved best practices and compensation actions between knowledge workers.

Knowledge workers are characterized as being knowledgeable in their business domain. Most often, this knowledge exists merely as tacit knowledge, so it is not explicitly available in an organization or company. For the proposed framework, this has two important consequences:

- The application of their knowledge must not be hindered by the IT system (i.e., a prescriptive approach must be avoided).
- This tacit knowledge is an opportunity to improve the enactment of cases in general (e.g., less experienced business users might benefit from formerly tacit knowledge of more experienced colleagues).

The proposed constraint framework aims at providing adequate support for these two implications. On the one hand, the framework does not intend to prohibit any action of knowledge workers. On the other hand, it is designed to support knowledge transfers through the ACM system to other knowledge workers. Figure 2 schematically shows the knowledge transfer cycle in the design of the framework. There exist three feedback loops that potentially cause the actions of knowledge workers to become influenced by other knowledge workers.

A. Recommendation Feedback Loop

In Figure 2, the loop starts with *Knowledge Workers* who handle cases (*Case Enactment*). The *Recommendation of Actions* component observes their actions continuously and learns from them. Eventually, *Knowledge Workers* can query *Recommendation of Actions* whenever they require advice. If a similar situation has occurred in the past, then a proposal for potential next actions is made. This loop directly feeds back the decisions of knowledge workers to other knowledge workers. Consequently, the function of the *Recommendation Feedback Loop* is the fast propagation of formerly implicit knowledge. This may include compensation actions to recover from a constraint violation and evolving practices how to handle specific situations.

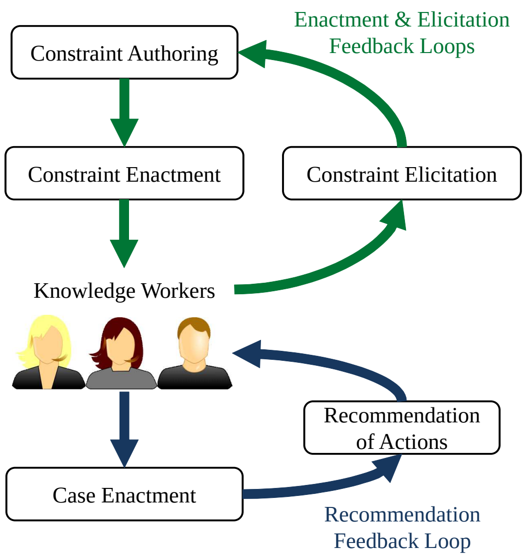


Fig. 2. Knowledge Transfer Cycle

B. *Enactment Feedback Loop*

This loop starts with *Knowledge Workers* who observe that a large quantity of *Constraint Enactments* of a specific constraint are in a state of violation. This might be an indication for changing this constraint, so *Constraint Elicitation* is performed to evaluate this particular constraint. If the set of constraints is adapted (*Constraint Authoring*), *Constraint Enactment* will propagate this change to *Knowledge Workers* during *Case Enactment*. Consequently, the main objective of the *Enactment Feedback Loop* is to react to constraint violations. If a large number of constraint instances with a particular constraint in a state of violation is observed, then there might be a need to adapt the set of constraints. Maybe the rules of the business have changed or there was an error introduced into the set of constraints unintentionally.

C. *Elicitation Feedback Loop*

This loop starts with *Knowledge Workers* who see the need for *Constraint Elicitation* (e.g., new constraints are needed to implement a compliance document which describes upcoming compliance requirements) and introduce new constraints by *Constraint Authoring* that become effective as constraint instances in *Constraint Enactment*. The *Case Enactment* by *Knowledge Workers* might be influenced by the changed set of constraints. In contrast to the *Enactment Feedback Loop*, this loop is not triggered by violations occurring during *Constraint Enactment*. Consequently, the main objective of the *Elicitation Feedback Loop* is to fully integrate knowledge workers directly in the constraint elicitation and creation process. Knowledge workers participate in constraint elicitation and actively contribute their domain knowledge while working on new internal policies that they formally specify as constraints. Once new policies are enacted as constraints of the ACM system, the knowledge workers' future decisions are potentially influenced by them.

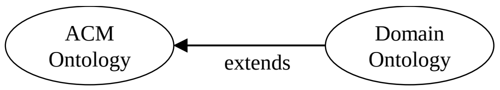


Fig. 3. Integration of domain-specific ontology with ACM ontology

III. Framework Components

A. *Ontology*

Ontologies are an efficient way to organize information. The set of components of ontologies include concepts, attributes, relations and instances. A domain ontology represents the knowledge of a specific business domain. Depending on the business domain, the ontology can look fairly different. For the ACM system, it is important that domain concepts are related to technical concepts of the ACM domain (Figure 3). Multiple business ontologies can be combined with this single ACM ontology. By this, the ACM system, for example, could be made aware that the domain concept 'Payment' is actually in technical terms a 'Task' (Figure 4). The example given in Figure 4 illustrates conceptually how the ontology of the domain can be connected to the ontology of the ACM system and how both are integrated in a constraint language (for illustration purpose also described by concepts and relations). Through this architecture, it becomes possible to define constraints on the basis of higher-level concepts. For example, there could be a policy "Shipped orders must eventually be paid" which can be represented in a textual constraint language as "Shipping is finished leads to Payment is finished". There are concepts derived from Payment, like Payment by Credit Card that are also covered by this constraint. Even a tighter integration of the business ontology is possible: Consider that not only concepts can be mapped to ACM elements but also larger structures of the domain ontology. For example, consider two concepts, 'Order' and 'Customer', and their relation 'can be placed by'. This would allow to derive a task 'Place Order' with performer 'Customer' directly from the domain ontology and further allows to use this domain knowledge not only during case enactment but also for the definition of constraints.

B. *Constraint Elicitation & Constraint Authoring*

The main purpose of *Constraint Elicitation* is the extraction of internal policies from compliance documents which will in further consequence be introduced to the ACM system as constraints, and the evaluation of existing constraints with regards to internal policies. In a next step, these policies are fed into the ACM system as constraints (*Constraint Authoring*).

Figure 5 illustrates *Constraint Authoring*. Knowledge workers can use the *Constraint Editor* to author constraints. Moreover, they can define the ontology of their business domain. A *Constraint Editor* provides the functionality to create *Natural Language Constraints*. These constraints consist of parts stemming from a *Constraint Grammar* that abstracts underlying formal verification techniques, and of other expressions that

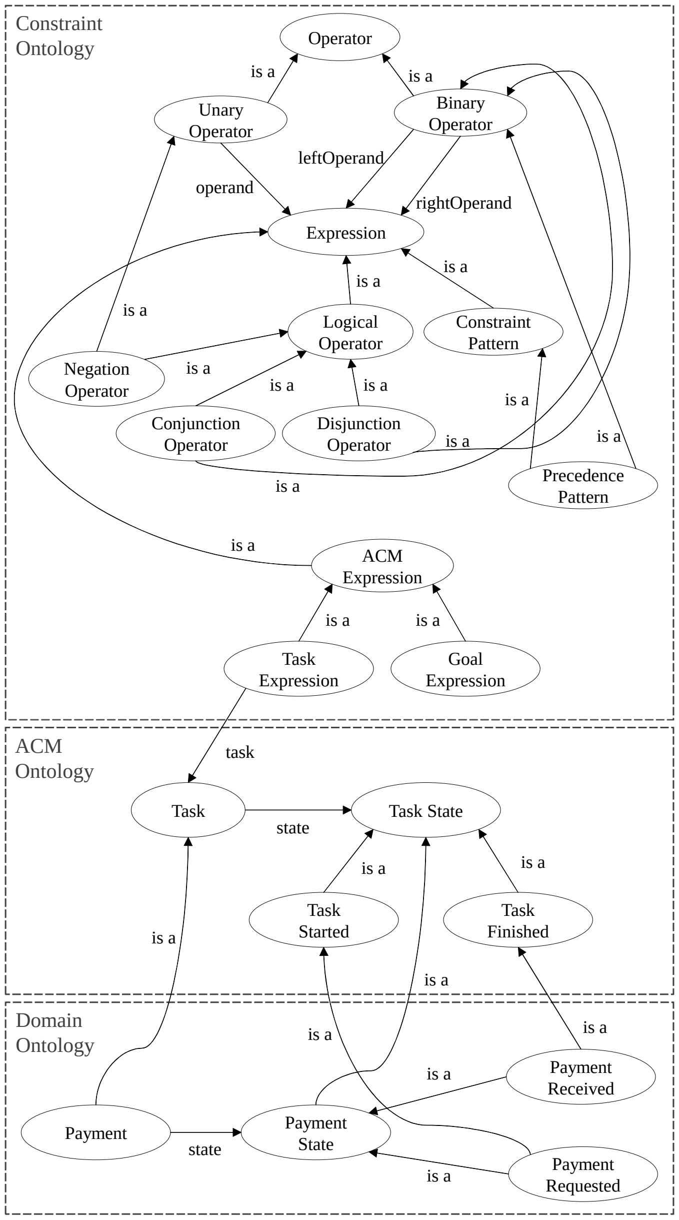


Fig. 4. Ontology Example

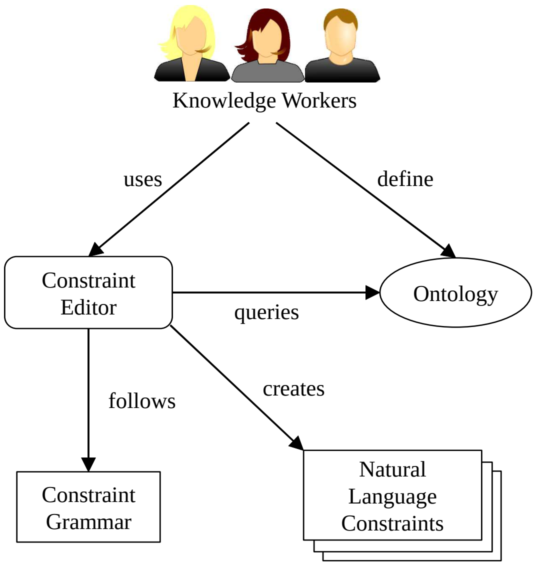
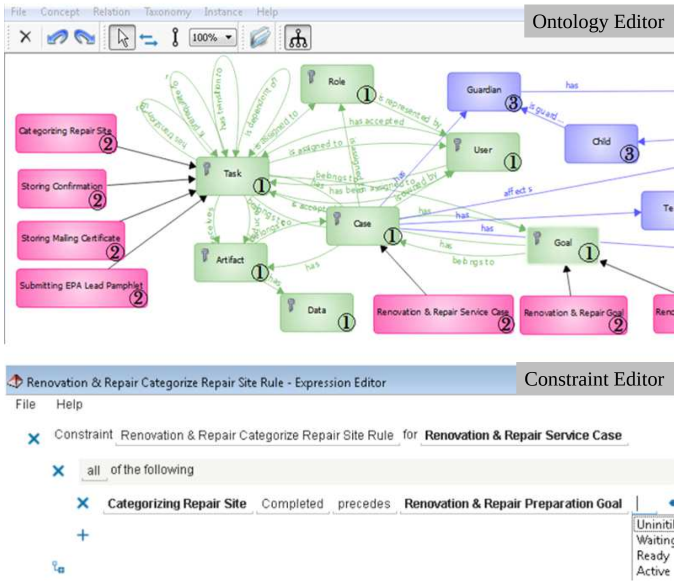


Fig. 5. Constraint Authoring



Renovation & Repair Categorize Repair Site Rule - Expression Editor
File Help
Constraint Renovation & Repair Categorize Repair Site Rule for Renovation & Repair Service Case
all of the following
Categorizing Repair Site Completed precedes Renovation & Repair Preparation Goal
Uninitil
Waiting
Ready
Active

Fig. 6. Screenshot of Constraint Editor and Ontology Editor

reflect the concepts and relations defined in the *Ontology*. A screenshot of our implementation of a user interface for editing constraints and the ontology is shown in Figure 6.

Since business users do not necessarily have a technical background, constraints should be easily understood. Consequently, technical constraint authoring must be avoided and a natural language approach is suggested. In the current state-of-the-art, the grammar or feature set of the language is rarely defined as part of the ontology (cf. [3]), but rather separate from it in a dedicated grammar that makes use of the ontology elements (cf. [4]). Either way, the core function of the domain ontology is the support of domain-specific concepts and relations, which are important not only for the definition of constraints but for the case enactment in general since knowledge workers operate based on their known domain concepts instead of technical terms.

Each created constraint has *five properties*. The first three properties are proposed by Leitner et al. [5] for flow-driven business processes. We adapt them to fit into the scope of ACM, discuss them in relation to our framework, and add an important fourth and fifth property named *Coverage* and *Regionality*:

(1) *Localization*: In which case instances must the constraint hold?

- Inter-Organizational Localization: The constraint is enacted for instances of cases that exist in more than a single organization. If the IT system is incoherent, the constraint must be either propagated to other organizations for enactment on their side, or the constraint is enacted centrally with the propagation of results to other organizations. Our framework assumes a single, coherent ACM system which can be used by several organizations (e.g., companies).

- Inter-Case-Concept Localization: The constraint is enacted for all instances of several cases. Hence, the constraint is imposed on multiple case concepts. This kind of localization requires a classification of a case instance to relate it to a case concept. This can be done either automatically (e.g., a specific case template is instantiated) or manually (e.g., the knowledge worker opens a case instance and decides for specific case concepts), depending on the circumstances. It is supported by the framework.
- Intra-Case-Concept Localization: The constraint is enacted for all instances of a specific case concept. It is supported by the framework.
- Intra-Case-Instance Localization: The constraint is enacted for a specific case instance only. This will be rarely the case since specialized compliance treatment of a case instance is rather unlikely. Consequently, Intra-Case-Instance Localization is not supported by the framework.

(2) *Span*: What happenings are observed for the enactment of the constraint?

- Intra-Case-Instance Span: The happenings of a single case instance are considered. That is, the constraint enactment does not observe happenings in other case instances for this constraint instance. This is the most common kind of span (cf. e.g., [6], [7], [8], [2], [9]). Since we have not yet encountered the requirement for other kinds of span in case studies (e.g., [8], [4]) as well, our solution supports this kind of span exclusively. Nevertheless, if a need for other kinds of span evolves, the proposed constraint support framework can consider the other kinds of span as well.
- Inter-Organizational Span: The happenings of all case instances of multiple case concepts that exist in multiple organizations are considered.
- Trans-Organizational Span: The happenings of all case instances of a single case concept that exists in multiple organizations are considered.
- Inter-Case-Concept Span: The happenings of all case instances of multiple case concepts are considered.
- Intra-Case-Concept Span: The happenings of all case instances of a case concept are considered.

(3) *Dependency*: A constraint is either independent of its previous enactments or dependent. If it is dependent, then it becomes, for example, possible to enact it only every second time. Since we have not yet discovered a practical use case for the *Dependency* property, it is not supported by the proposed framework.

(4) *Coverage*: A constraint covers specific periods of time. There exist two periods:

- Enactment Period: The constraint is enacted in this period of time. If the constraint is not yet active (current time $<$ enactment start time) or already expired (current time $>$ enactment end time), the constraint enactment component does not consider it.
- Retrospect Period: Past events, which occurred before the constraint entered the enactment period, can have an influence on the current state of a constraint. Thus, constraint enactment must also take these happenings into account for the defined period of time. Obviously, the retrospect period ends with the start of the enactment period. The retrospect period requires the ACM system to provide happenings falling into this period of time.

(5) *Regionality*: If the organization is (or the organizations are) spread over different regions (e.g., different countries), each region might have different regulations that must be met. Additionally, there might be cultural differences in the way how cases are handled by knowledge workers. Our approach considers this by allowing the assignment of constraints to specific case concepts for different regions that are derived from more general case concepts.

C. Case Enactment

Case Enactment is the basic functionality of every ACM system. Knowledge worker collaboratively work towards goals by performing tasks, and data and content (e.g., documents) are continuously modified. This component must capture the state and actions of a case extensively. Only then is the constraint framework working effectively. If knowledge workers perform actions that are not disclosed to the IT system, then Constraint Enactment possibly cannot provide notifications regarding violations, and knowledge transfers through the Recommendation of Actions would not work. It is important to raise the awareness of knowledge workers to properly document all of their (manual) actions.

D. Constraint Enactment

There exist various possibilities for enabling *Constraint Enactment*. Linear Temporal Logic (LTL) is an established formal way to describe specifications for both design time (cf. [10], [11]) and runtime verification (cf. [6], [12]). Dwyer et al. propose a set of property specification patterns that abstract temporal logic formulas to high-level order and occurrence patterns. Elgammal et al. build upon this pattern set and create a compliance language [2]. Temporal patterns can be represented by different underlying formalisms and checking techniques. Complex Event Processing (CEP) can process a large quantity of events in close to real-time and is applied for temporal pattern-based runtime verification of business processes in recent studies ([8], [7]). Our constraint enactment approach and prototypical implementation is based on CEP. For a detailed survey on approaches and categorization based on functionalities, we refer the interested reader to the Compliance Monitoring Framework (CMF) as proposed by Ly et al. [13]. They analyze existing compliance monitoring approaches based on ten Compliance Monitoring Functionalities (CMFs):

- CMF 1: Constraints referring to time
- CMF 2: Constraints referring to data
- CMF 3: Constraints referring to resources
- CMF 4: Supporting non-atomic activities
- CMF 5: Supporting activity life cycles
- CMF 6: Supporting multiple instances constraints
- CMF 7: Ability to reactively detect and manage violations
- CMF 8: Ability to pro-actively manage violations
- CMF 9: Ability to explain the root cause of a violation
- CMF 10: Ability to quantify the degree of compliance

The CMF framework does not yet contain criteria to categorize the support for constraint authoring and maintainability, or the integration of ontologies. Many existing compliance monitoring approaches still require well-trained, rather technical personnel for creating and maintaining constraints, which is still an obstacle for the practical adoption in real-world applications. Shifting the language of constraint specification towards natural and domain specific terminology might enable non-technical business users to actively take part in managing constraints.

E. Recommendation of Actions

As Ly et al. point out in [13], so far there exists no compliance monitoring approach which supports more than seven of their identified ten CMFs. To illustrate the challenges involved with supporting specific existing CMF combinations, let us consider the combination of CMF 2 (data) and CMF 8 (pro-active management). To the best of our knowledge, none of the existing approaches provides support for this combination. Existing approaches realize pro-active support by using Constraint Programming (CP) to plan a case execution by finding solutions that satisfy all constraints eventually (cf. [14], [15]). It is hardly surprising that these approaches focus on the order of tasks because this level of abstraction allows to model an optimization problem of a reasonable size (assuming that the number of task is not very large). Existing approaches work as follows: The start and completion events of tasks are represented by an integer value and the optimization tries to find an optimal order for these events given an objective function and a set of constraints. For example, objective functions may minimize time or costs, and a constraint may demand that a variable is smaller than another variable to represent common compliance patterns, such as Precedence or Response. If data were included in this optimization process, there would be a time and a specific value for each data adaption. Consequently, the optimization problem would become infeasible to solve. Moreover, planning specific future data changes seems like a pointless exercise because it simply cannot be planned beforehand. In ACM, data is an essential part of every case, so constraints that involve data are likely to occur, but existing pro-active compliance support approaches are not able to support data-based constraints.

To overcome this limitation, we propose to leverage the decisions made by knowledge workers for the automated recommendation of next actions. The *Recommendation of Actions* component seeks to leverage past actions of knowledge workers to learn from their decisions and to subsequently provide support for other knowledge workers who are in similar situations. Tran et al. propose a *User Trained Agent (UTA)* for the recommendation of actions [16]. The current functionality of the UTA is as follows: Every time a knowledge worker adds an ad-hoc task to a case, a training sample is generated. Once there are several trainings samples under a specific goal collected, features are selected [17]. Based on these features, a clustering decision tree is generated [18]. The UTA preprocesses data to some extent but does not yet consider temporal relationships that might have an influence on the decisions of knowledge workers. Moreover, it does not yet sufficiently integrate knowledge stemming from the ontology and constraints. We are currently working on a prototypical extension of the UTA that aims at improving the preprocessing capabilities while preserving a reasonable computational complexity.

The *Recommendation of Actions* component seeks to learn from actions of knowledge workers and to propagate the learned knowledge to other knowledge workers. This can be achieved by machine learning after having prepared the inputs for the learning process properly. Figure 7 illustrates the learning process. The enactment of cases by knowledge workers is recorded as *Execution Logs*. *Selection* is a filtering process to consider only those execution logs that are needed to create a prediction model with a specific purpose (e.g., to help compensating specific compliance violation). By *Preprocessing*, the *Selected Execution Logs* are prepared for *Machine Learning* as *Training Samples*. Finally, a machine learning approach creates a *Prediction Model* on basis of the provided training samples.

Figure 8 shows how suggestions for next actions are made. A knowledge worker requests a *Next Action Suggestion* for a *Case Instance* that she or he is working on. Each enacted case instance pertains information of all happenings in this instance in an *Execution Log*. *Preprocessing* prepares the data of the log as inputs for the instantiation of a *Prediction Model*. The *Prediction Model Instance* contains probabilities for performing specific next actions.

1) Selection: An important aspect of preprocessing is the selection of execution logs. For example, if a knowledge worker seeks help to compensate a compliance violation, only those execution logs might be included for learning decisions where this compliance violation was successfully resolved. If the current case execution is compliant, it could be harmful to include execution logs into the learning process that could lead to non-compliance.

2) Preprocessing: A log must contain all the information that might have an influence on the decisions of knowledge workers. That includes events related to activities or data adaption with meta-data such as the performer and timestamp of the action as well as information about constraint violations.

The preprocessing component prepares the raw data from logs for the automated learning process. To improve the learning process, we propose to include temporal aspects of decision (cf. [19], [20]) into the learning process. Moreover,

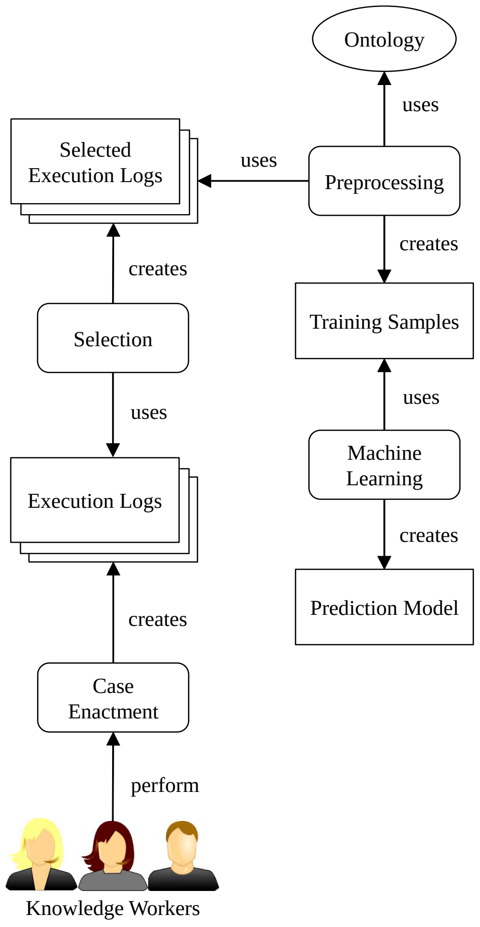


Fig. 7. Learning from the actions of knowledge workers

we propose the integration of domain knowledge given by the ontology and the states of constraints. This involves diverse ways to extract useful data, such as:

- Relative Intra-Instant Data Consideration: A relative or aggregated value is created from two or more data values for the same instant in time. For example, the account balance is computed from incoming and outgoing payments.
- Relative Inter-Instant State Preprocessing: The relative time between states of task, goals, events and constraints, such as the start or end time is calculated. For example, the time between the end of a medical examination and the start of a surgery is computed.
- Relative Inter-Instant Data Preprocessing: The relative change of data values from one instant in time to another instant in time is computed. For example, the difference of temperature data is computed from its values at instant t-1 and instant t.
- Absolute Inter-Instant Data Preprocessing: Not only relative but also a series of absolute values of the same data in different instants in time can be useful. For example: The temperature of a patient is above 39 degrees Celsius for a longer time which causes a special decision of a knowledge worker.

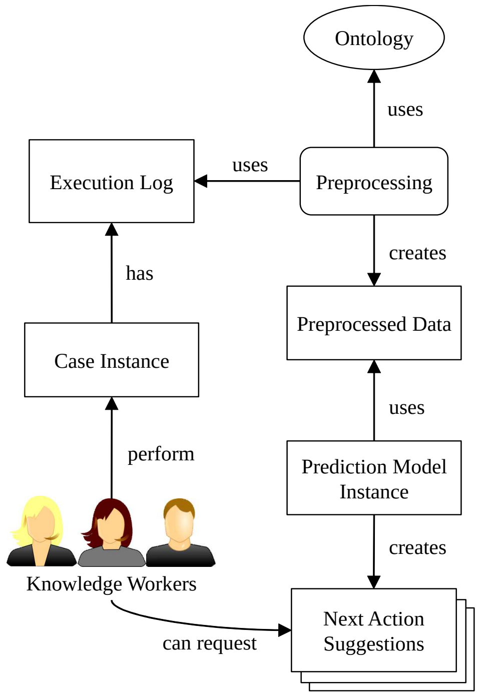


Fig. 8. Suggesting next actions

The more information is prepared by this preprocessing, the bigger is the resulting learning problem. Consequently, it must be carefully decided on how to enrich the provided raw data for the learning process:

- By having domain knowledge available from the ontology, preprocessing may focus on concepts which are related to each other.
- Temporal windows may be used to focus on happenings that date back a specific amount of time and to abstract from events having happened earlier.

3) Machine Learning: Automated learning from the decisions of knowledge workers is realized by the supervised learning approach *classification*. Based on the classifiers learned from the training data, new observations can be assigned to the category that they belong to. Categories are in the context of this framework equivalent to tasks, so that it can provide recommendations for next actions. An observation is equivalent to the state of the case execution for which the knowledge worker requests a proposal and must in the same way be preprocessed as the training samples to create a matching set of attributes.

IV. Discussion

We present the framework in the context of ACM, but flexible business process management approaches can benefit from the framework in general. In such systems, a fast way to react to changing compliance requirements is needed, which can be provided by the proposed ontology-based constraint editor. Additionally, an automated support for deciding on next actions can be beneficial when business users are not confined to enacting predefined, flow-driven business processes.

To make tacit knowledge explicit in form of constraints, it would be possible to extend the proposed framework by constraint discovery. By supporting the automated discovery of constraints, the framework would gain an additional feedback loop. The main objective of this loop would be the capturing of recurring behavior that is existent in a large quantity of case instances but not yet available as explicit constraint. Current constraint discovery research is predominantly focused on discovering LTL-based temporal patterns [9]. Another approach mines pattern-based organizational constraints for the DPIL approach [21]. The integration of domain knowledge stemming from an ontology is not yet considered in existing approaches. Taxonomies and relations might be of great use to improve constraint mining results and to make mined constraints better understandable for business users.

Lakshmanan et al. propose a markov prediction model for data-driven semi-structured business processes [22]. By exploiting process mining techniques, the approach discovers a classical process model to learn decisions for determined decision points. If the recommendation approach of the proposed framework can benefit from this additional preprocessing step is uncertain since classical process mining tends to create spaghetti models when analyzing enactments of unstructured business processes, such as those often present in ACM.

V. Conclusion & Future Work

This paper proposes a framework that supports knowledge worker to avoid non-compliance. An ontology-based constraint editor allows knowledge workers to rapidly react to new circumstances that require an adaption of the realization of compliance requirements by enabling the creation of constraints in business terminology. Consequently, long maintenance cycles—usually involved with realizing compliance requirements in an IT system—are avoided. Decisions of knowledge workers, such as those related to compensating a compliance violation, are being automatically learned to provide support to knowledge workers that encounter a similar situation.

There exist several opportunities for future research. Knowledge workers might benefit from extending the framework with automated constraint discovery. How to integrate the process of discovery with the ontology and how to make the discovered results usable by knowledge workers would be an interesting direction. User studies on specific aspects of the framework, such as the usability of the constraint and ontology editor, could be used to further investigate the practical applicability of the framework.

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] 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.

[3] J. Yu, T. P. Manh, J. Han, Y. Jin, Y. Han, and J. Wang, "Pattern based property specification and verification for service composition," in *7th International Conference on Web Information Systems*. Springer, 2006, pp. 156–168.

[4] T. Tran, E. Weiss, C. Ruhsam, C. Czepa, H. Tran, and U. Zdun, "Embracing process compliance and flexibility through behavioral consistency checking in acm: A repair service management case," in *AdaptiveCM'15*, ser. Business Process Management Workshops 2015, August 2015.

[5] M. Leitner, J. Mangler, and S. Rinderle-Ma, "Definition and enactment of instance-spanning process constraints," in *International Conference of Web Information System Engineering*, ser. LNCS. Cyprus: Springer, 2012, pp. 652–658.

[6] 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.

[7] A. Awad, A. Barnawi, A. Elgammal, R. Elshawi, A. Almalaise, and S. Sakr, "Runtime detection of business process compliance violations: An approach based on anti patterns," in *30th Symposium on Applied Computing*, ser. SAC'15. ACM, 2015, pp. 1203–1210.

[8] T. Tran, E. Weiss, C. Ruhsam, C. Czepa, H. Tran, and U. Zdun, "Enabling flexibility of business processes by compliance rules - a case study from the insurance industry," in *BPM'15 (Industry Track)), Innsbruck, Austria, September 2015.*, 2015, pp. 30–43.

[9] C. D. Ciccio and M. Mecella, "On the discovery of declarative control flows for artful processes," *ACM Trans. Manage. Inf. Syst.*, vol. 5, no. 4, pp. 24:1–24:37, Jan. 2015.

[10] A. Cimatti, E. Clarke, F. Giunchiglia, and M. Roveri, "NuSMV: a new Symbolic Model Verifier," in *11th Conf.on Computer-Aided Verification (CAV)*. Springer, July 1999, pp. 495–499.

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

[12] W. M. P. van der Aalst and M. Pesic, "DecSerFlow: Towards a truly declarative service flow language," in *3rd International Conference on Web Services and Formal Methods (WS-FM)*. Springer, 2006, pp. 1–23.

[13] 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.

[14] I. Barba, B. Weber, C. D. Valle, and A. Jimnez-Ramrez, "User recommendations for the optimized execution of business processes," *Data and Knowledge Engineering*, vol. 86, no. 0, pp. 61 – 84, 2013.

[15] M. T. Gómez-López, L. Parody, R. M. Gasca, and S. Rinderle-Ma, *OTM 2014 Conferences, Amantea, Italy, October 27-31*. Springer, 2014, ch. Prognosing the Compliance of Declarative Business Processes Using Event Trace Robustness, pp. 327–344.

[16] T. Tran, C. Ruhsam, M. J. Pucher, M. Kobler, and J. Mendling, "Towards a pattern recognition approach for transferring knowledge in acm," in *AdaptiveCM'14*, 2014.

[17] F. Fleuret, "Fast binary feature selection with conditional mutual information," *J. Mach. Learn. Res.*, vol. 5, pp. 1531–1555, Dec. 2004.

[18] H. Blockeel, L. D. Raedt, and J. Ramon, "Top-down induction of clustering trees," ser. ICML '98. Morgan Kaufmann Publishers Inc., 1998, pp. 55–63.

[19] M. M. Akhlagh, S. C. Tan, and F. Khak, "Temporal data classification and rule extraction using a probabilistic decision tree," in *ICCIS'12*, vol. 1, June 2012, pp. 346–351.

[20] Q. Shi, Y. Zhao, and M. Liu, "Towards learning segmented temporal sequences: A decision tree approach," in *ICMLC'15*, vol. 1, July 2015, pp. 145–150.

[21] S. Schönig, C. Cabanillas, S. Jablonski, and J. Mendling, "Mining the organisational perspective in agile business processes," in *BPMDS'15, Stockholm, Sweden, June 8-9*, 2015, pp. 37–52.

[22] G. Lakshmanan, D. Shamsi, Y. Doganata, M. Unuvar, and R. Khalaf, "A markov prediction model for data-driven semi-structured business processes," *Knowledge and Information Systems*, pp. 1–30, 2013.