Embracing Process Compliance and Flexibility through Behavioral Consistency Checking in ACM

A Repair Service Management Case

Thanh Tran Thi Kim[1], Erhard Weiss[1], Christoph Ruhsam[1]

Christoph Czepa[2], Huy Tran[2], Uwe Zdun[2]

[1] ISIS Papyrus Europe AG, Maria Enzersdorf, Austria
{thanh.tran,erhard.weiss,christoph.ruhsam}@isis-papyrus.com
[2] Research Group Software Architecture, University of Vienna, Austria
{christoph.czepa,huy.tran,uwe.zdun}@univie.ac.at

Abstract. Enabling flexibility in unpredictable situations with ad hoc actions decided at runtime by knowledge workers is the main focus of Adaptive Case Management (ACM) systems. However, ad hoc actions added during case execution and ACM templates prepared at design time need to be within the boundaries defined by business constraints, company regulations and legal systems. In this paper we report our experience in addressing this challenge by using model checking and runtime monitoring techniques for behavioral consistency checking that can handle both ACM aspects: support by means of predefined process templates and high flexibility by allowing ad hoc actions at runtime. Our study is conducted using a practical ACM system for repair service management handling different customer requirements under diverse compliance and law regulations.

Keywords: Adaptive Case Management, ISIS Papyrus ACM solution, Consistency Checking, Compliance Checking, Business Process Management, Compliance Rules

1 Introduction

A challenge of business process management (BPM) systems is to react on spontaneous or short time process adaptions due to changing business conditions. The rigidness of process definitions and the involved bureaucracy with change management cycles are hindering process innovation. Although many different approaches have been suggested to enhance the flexibility of BPM systems, most of them have not managed to escape from the restrictive nature of the approach [1, 2]. ACM is proposed in the context of knowledge intensive and content centric work. The ACM principle "design-by-doing" allows users acting as knowledge workers (KW) to evolve processes embracing both, support of predictable business scenarios by using

well-defined sub-processes and unpredictable scenarios by allowing ad hoc actions and changes during runtime.

A rigid process configuration aims to ensure that business operation is under control and satisfies the compliance requirements related to law and regulations. Thus, process validation needs to consider both, the process model (syntactical consistency) and the compliance requirements (semantic consistency) [3]. As a consequence of the evolving nature of ACM cases with ad hoc actions it is unclear how to ensure the process compliance within the constraints imposed on the business operation by compliance requirements. There are numerous techniques for checking and handling inconsistencies for well-defined business processes [4].

In our study, the compliance requirements are described using temporal logics, represented in terms of high-level specification patterns [5]. As the KWs involved in our studies found temporal logics unfamiliar to their domain of expertise, we devise a simple high-level domain-specific constraint language that is close to natural language to help the KWs in describing compliance requirements defined as compliance rules. These are automatically translated into temporal logics to be verified for any potential inconsistencies by powerful model checkers and runtime monitoring engines. We report in this paper the first attempt in applying well-known formal modeling and verification techniques to support KWs in ensuring business compliance in ACM. Our experience is based on applying consistency checking techniques to an ACM solution for repair service management (RSM) in the ISIS Papyrus ACM system [6]. In RSM, diverse requirements from a broad variety of customers have to comply with a vast range of different laws and regulations. The multiplicity of possible requirements, constraints and exceptions cannot be efficiently covered alone by predefined processes. Depending on particular cases, the repair service officer, i.e. the KW, decides on specific conditions and work lists corresponding to the requirements of the customer and the discovered circumstances. Still, the KW needs to act in compliance defined by company rules and laws because violations permanently jeopardize the business and its economical reputation. Since rules must be considered at runtime and design time, compliance checking must apply for ad hoc actions as well as templates.

Our work is related to existing studies in the domain of verification of business processes at design or runtime and to specification languages for the definition of business rules. Liu et al. propose a framework for model checking of business processes at design time that transforms process models into Finite State Machines (FSM) and translates constraints from a graphical specification language into LTL [7]. Namiri et al. translate temporal logic patterns to Event-Condition-Action rules and perform runtime verification [8]. An approach presented by Awad et al. aims at checking compliance of business process models using a visual query language BPMN-Q with recurring patterns like ‘leads to’ and ‘precedes’ to describe constraints and performing model checking to assure constraints are satisfied [9]. Van der Aalst et al. proposes an approach for checking LTL properties against an event log [10]. Complex Event Processing (CEP) is used for compliance monitoring in the context of service-oriented architectures (SOA) [11, 12]. The vast amount of existing studies in this field affirms the need for research and contributions, especially in the area of

runtime verification [13]. We are neither aware of previous studies that proposed a unified checking approach based on model checking and CEP for the verification of ACM cases at runtime and design time nor of existing experience reports on handling business compliance in the context of ACM.

The remainder of our paper is organized as follows: Section 2 introduces our concept for consistency checking as applied to the ISIS Papyrus ACM system. The configuration process for behavioral consistency checking is described in Section 3. Section 4 describes our experience of applying the checking in the RSM application. Finally, Section 5 discusses the lessons learned and concludes the paper.

2 Concepts of Consistency Checking in ACM

2.1 Overview of ACM

ACM cases are driven by goals which are in combination with tasks the core ACM elements. A significant feature of ACM systems is that the main actor, i.e. the KW, is empowered to handle events that are unpredictable or even never happened before. KWs use their implicit knowledge stemming from experience as well as from assessments and research of the current situation to react flexibly to new requirements in daily work within a permanently evolving business environment. To support KWs in such situations, ACM provides (a) automation of routine work by sub-processes which are predefined at design time and executed in the defined order and (b) flexibility by the free design of activities and their sequences through ad hoc goals and tasks which are added at runtime by the KW [14, 15]. Goals are achieved by completing its sub-processes and/or ad hoc tasks and a case closes when all its goals are reached.

2.2 Consistency Checking Applied to the ISIS Papyrus ACM system

Figure 1 shows an overview of the approach taken in this paper: The ISIS Papyrus ACM system is enhanced with a consistency checking system to ensure that the ACM case definition at design time and its evolution at runtime do not violate structural and/or behavioral consistency.

Consistency is regulated by a set of constraints formulated by compliance rules stored in a rule collection, which are expressed in natural language by business administrators. At design time, business administrators define goal and task templates, and link them with related compliance rules. Consistency checking is done in a two-fold way: (a) The structural and behavioral consistency of templates is verified by using model checking alerting the administrator in case of violations. As model checking is computationally expensive and demands well-defined case models, it suits a design time approach. (b) The focus of this paper is on compliance checking at runtime, when KWs create ad hoc actions to reach a goal. These are verified by on-the-fly checking, which operates on the current execution trace of a running case instance and is based on CEP [16] providing real time feedback in case of constraint violations by alerting KWs performing ad hoc changes at runtime.

As shown in Figure 2, compliance rules from different sources are imposed on the execution of ACM cases which are related to laws, contracts with business partners, general standards, best practices and company internal regulations. A usually underestimated source is coming from tacit business knowledge of the KWs [17].

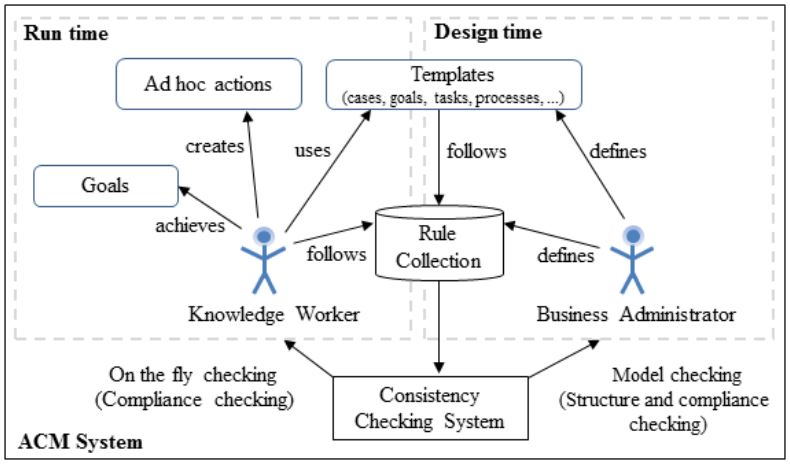

Fig. 1. The ISIS Papyrus ACM environment with consistency checking

In real life, there are basically two scenarios where compliance rules have to influence the case execution:

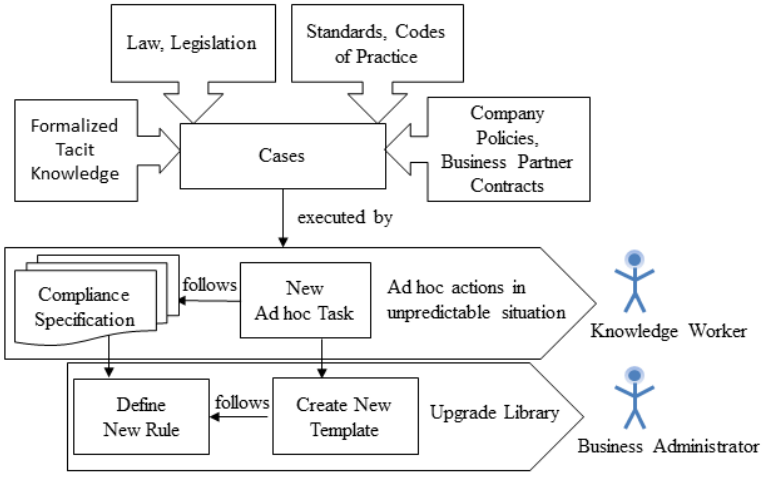

Fig. 2. Definition and maintenance of compliance rules in ACM systems

1. Unpredictable situations: (1.1) In such cases, there are typically not yet suitable templates available. Thus, the KW creates ad hoc actions from generic templates and takes the responsibility that the ad hoc actions comply with constraints, that

must be followed [18]. (1.2) Applying the best practice principle, business administrators and/or empowered KWs create new templates from case instances, which proofed that case goals where repeatedly satisfyingly reached. Additionally, new compliance rules, which were discovered in (1.1) will be added to the rule collection. This way, knowledge is preserved in the system to support KWs the next time when such situations occur.

2. Predictable scenarios: Alternatively, new situations, which a company plans to deal with, can be analyzed and prepared in advance by business administrators. The necessary templates and associated compliance rules are added to the system similar to (1.2) above, before KWs will start to work on such cases.

In both (1.2) and (2) the business administrators benefit from compliance checking as they will receive alerts during design time if certain constraints are violated.

We have successfully applied a User Trained Agent [19] to suggest best next actions to KWs in ad hoc situations that are similar to previously handled cases. In the context of behavioral consistency checking it is important that the UTA builds its knowledge on compliant ad hoc actions. Thus, the definition of compliance rules is important in the maintenance of ACM systems.

3 Configuration Process for Behavioral Consistency Checking

Figure 3 shows a configuration process for applying the behavioral consistency checking in ACM. The process includes two phases: (a) translating the compliance specifications stemming from different sources from natural language into compliance rules expressed in a constraint language, (b) assigning compliance rules to templates.

Allowing business people to define such rules in natural language, a constraint rule editor is supplied, supporting business terms defined by business domain specific ontologies. It is built upon temporal logic patterns to support the definition of consistency requirements [20] [5]. The rules are linked to ACM elements which are mapped by business domain specific ontologies to the underlying ACM object model. This way, new templates (goals, tasks) being adjusted to new requirements are affected by existent compliance rules, as they are mapped with the associated ontology concepts. Depending on the scope of rules, they are stored in a rule collection for later reuse in several templates or embedded into specific cases. Compliance rules defined within our study are classified into two types which are created for different situations:

(a) State-based rules define the sequence of actions purely based on the state of tasks or goals. For example *TaskB* must start after completion of *TaskA*.

(b) Data-based rules explicitly involve data in the rule definition. An example is *TaskA.Attribute_n > value*. If a new rule relates to data that is already defined in the system, business administrators simply create a new data-based rule without support from database experts.

However, if the needed data (*Attribute_n* in the example above) are not yet modeled in the system, database experts would be needed for the implementation of the new data requirement with all agility hindering consequences of a product release

management cycle. To overcome such situations, state-based rules in combination with voting tasks can be defined to enforce the necessary execution sequence. Voting tasks contain checklists (see Figure 5a) that are suitable for checking conditions manually by users. Later it may be worthwhile to add such data from best practice cases explicitly to the data model and apply them in data-based rules.

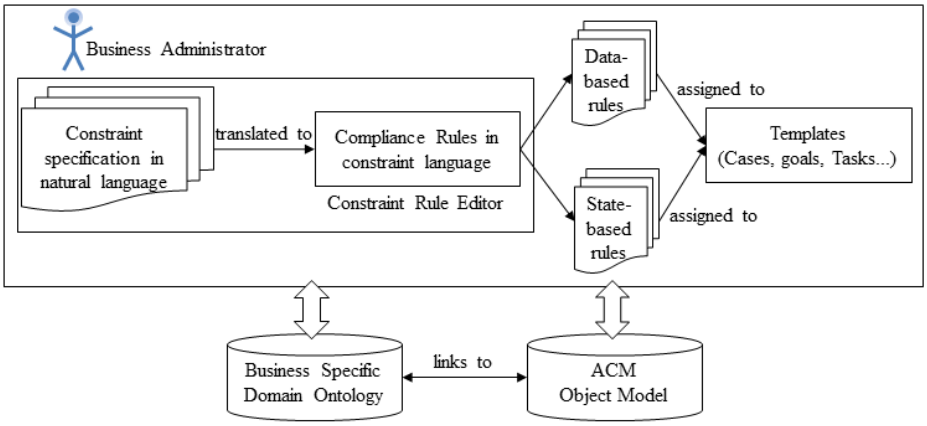

Fig. 3. Configuration process for behavioral consistency checking in ACM systems

A constraint is composed from business ontology elements which relate to the ACM core elements case, goal, task, data, and temporal patterns expressing the temporal dependency of them, e.g.:

```
Constraint C1 for CaseN{
  TaskA.finished precedes TaskB.started}
```

A constraint can be assigned to specific cases (*Constraint C1 for CaseN*) which will be triggered when the defined elements in that case are affected, or defined globally without the 'for'-part, which will be triggered if any of the defined elements are affected. The states of ACM elements, considered in the constraint definitions are for goals {initiated, reached, ongoing} and tasks {started, finished, running}.

To enable the definition of more detailed temporal patterns [5], we introduce the following constraint language expressions:

- Existence: *TaskA.finished* occurs
- Absence: *TaskA.finished* never occurs
- Response: *TaskA.finished* leads to *TaskB.started*. It means if *TaskA.finished* happens, *TaskB.started* must follow.
- Precedence: *TaskA.finished* precedes *TaskB.started*. It means if *TaskB.started* shall happen, *TaskA.finished* must happen first.

Additionally, each pattern of the aforementioned constraint language expressions can have a scope, which defines the temporal extent for constraint observation. For example, if the constraint *C1* shall apply only after *TaskA.started*, it is defined as

```
Constraint C1 for CaseN{
After TaskA.started
    [TaskA.finished precedes TaskB.started]}
```

The scope *after* defines the execution after a given event. Particularly in this example, only after the event *TaskA.started* happened, the events *TaskA.finished precedes TaskB.started* must be executed. As shown in Figure 4, there are five basic kinds of scopes: (1) Global: the entire execution; (2) Before: before the first occurrence of event R; (3) After: after the first occurrence of event R; (4) Between: between events Q and R; (5) After-Until: after event Q until event R.

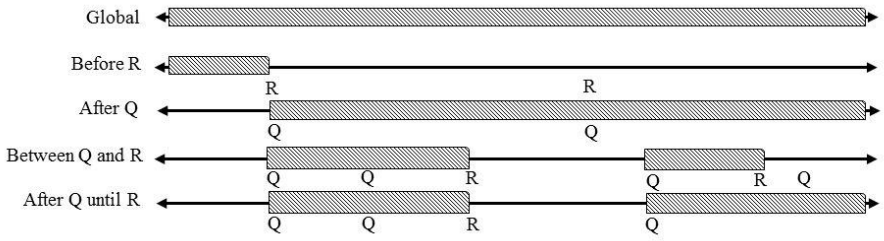

Fig. 4. Pattern scopes

4 Repair Service Management

ISIS Papyrus delivers business applications serving cross industry customer requirements. The RSM application is based on the requirements of a customer from the managed services domain offering facility repair services to industrial and commercial customers. It was used as prototype for applying the consistency checking solution and validating its practicability. These services involve maintenance, cleaning and repairs performed by the enterprise's own employees as well as the engagement of subcontractors. A broad variety of requirements from the customers and a large quantity of different laws and regulations must be considered.

From our experience we notice that several business domains are rather overregulated. Thus, compliance regulations imposed externally must be translated into internal directives being obeyed by processes and KWs and also support external audit requirements. Company internal standards are treated similarly although they may be executed more casually. If quantitative measures like time or money are defined, KWs frequently look for workarounds to reach their goals faster. If they are instructed to follow strictly predefined steps their knowledge from daily work is not considered. Thus, excluding their experience from process adaptions causes frustration and poor customer experience.

The RSM case focuses on the on-the-fly checking as well as on the possibility to perform consistency checking on ACM templates prepared by business administrators. We focus on the compliance considerations implied by the United States Envi-

ronmental Protection Agency's Lead Renovation, Repair and Painting Rule (RRP Rule) [21] and the consistency of the team selection process.

4.1 Compliance Definitions

A repair service officer, acting as KW of a company supplying repair services, receives an order for the replacement of several water pipes in a staff accommodation of the customer in the United States of America. According to the RRP Rule, the KW who is dealing with a new service request needs to check whether the repair service involves activities disturbing painted surfaces in a home or child-occupied facility built before 1978. Then, it requires a certified inspector or risk assessor to confirm the repair site to be free of lead-based paint or itemize and depict the surfaces covered with lead-based painting.

Following the scenario (1.1) from section 2.2, the KW creates a new case for this repair service order and when facing RRP rules for the first time, adds one or more generic ad hoc tasks for the treatment of the RRP rule related examinations. He has to check, (a) whether painted surfaces will be disturbed; (b) whether the repair site is in a home or child-occupied facility; (c) whether the repair site is in a building built before 1978; (d) whether the repair site contains lead-based paint (i.e. whether the repair site has been inspected by a certified inspector or needs to be inspected for objects painted with lead-based paint); (e) whether a list of items and surfaces covered with lead-based paint exists or needs to be compiled and whether the repair activities will affect one of this items and surfaces.

Based on the outcome of the examinations, the KW continues with the case, executing ad hoc tasks selected from a list of specific or generic task templates. When the case handling was confirmed as best practice, an empowered KW or business administrator saves related tasks as templates and the business administrator builds specific task templates for the investigation tasks containing checklists (see Figure 5a), composes the necessary compliance rules and references the rules to the repair service case template.

The checking result is implied through a voting by the KW: approved for *Yes* and deny for *No*. The *TaskRRP_Repair* contains a checklist proving whether painted surfaces will be disturbed by the repair activities or not. The *TaskRRP_Site* holds a checklist with the information whether the repair site is a home or a child-occupied facility and built before 1978. The *TaskRRP_Lead* contains a checklist informing on the inspection status of the repair site. The *TaskLead_Items* holds the information whether a list of items and surfaces covered with lead-based paint exists or needs to be compiled and whether the repair service affects one of those items and surfaces. These task templates are involved in the RRP rule which is described in the following compliance rules.

```
Constraint RRP_Compliance0 for RepairServiceCase {
  TaskRRP_Repair.started occurs exactly 1x}
Constraint RRP_Compliance1 for RepairServiceCase {
  TaskRRP_Repair.approved leads to TaskRRP_Site.started}
Constraint RRP_Compliance2 for RepairServiceCase {
```

```
  TaskRRP_Site.approved leads to TaskRRP_Lead.started}
Constraint RRP_Compliance3 for RepairServiceCase {
  TaskRRP_Lead.approved leads to TaskLead_Items.started}
```

Compliance rules are checked on the fly during the execution of ad hoc actions. For the definition of compliance rules in natural language, a constraint rule editor implemented in the ISIS Papyrus ACM system (see Figure 5b) supports business administrators with terms used in their daily business work. These are concepts defined in a business domain specific ontology. The task templates *TaskRRP_Repair* and *TaskRRP_Site* are mapped to business language through the ontology concepts *Repair Impact* and *Site Categorization*, respectively.

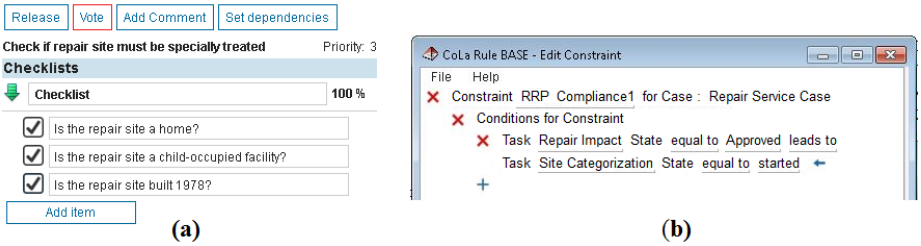

Fig. 5. *Task_RRP_Site* with checklist and constraint rule editor in ISIS ACM system

4.2 On-the-fly Consistency Checking

The next time such an RSM case is executed making use of the newly created templates, the first rule is triggered on case instantiation fulfilling *RRP_Compliance0*. This means, *TaskRRP_Repair* is part of the case and thus, cannot be skipped or added multiple times. The KW needs to check on the issue and fulfills the task by selecting a vote. If *TaskRRP_Repair* is approved, *TaskRRP_Site* is suggested to the clerk to end the temporary violation of constraint *RRP_Compliance1*. As soon as *TaskRRP_Site* is approved, *TaskRRP_Lead* is suggested to the clerk to end the temporary violation of constraint *RRP_Compliance2*, and so on.

Although the sequence of tasks is constrained in this case, other tasks can be executed in between which empowers the KW to react to the specific situation as needed. For example, after the completion of *TaskRRP_Repair*, the clerk could add the tasks *Order Material*, *Choose Workers*, and so on. At that stage of *TaskRRP_Repair* the constraint *RRS_Compliance1* is still temporarily violated as *TaskRRP_Site* is not started yet. The KW is notified of the violation by proposing it as next task. But he has the flexibility to perform any additional actions, which he assesses are necessary to fulfill the goal of the case. The proper point in time for the execution of the suggested *TaskRRP_Site* is determined by the KW.

Every action of the clerk triggers events monitored by the online checker. The online checker collates the information of the events with the constraints of the compliance rules. If rules are violated, the online checker produces notifications and suggestions. By creating state-based rules with the compliance rules editor and building

related task templates with checklists and voting options, the business administrator is able to induce consistency checking without the involvement of IT development. This approach allows flexible and fast adaption of rapidly changing rules through KWs without unnecessarily restricting the adaptability of ACM due to changes of underlying data models by IT.

5 Lessons Learned and Conclusion

Our general finding in this study is that behavioral consistency checking of compliance rules has significantly enhanced the ACM solution by embracing flexibility and enforcing control of company regulations or laws where imposed. Adequate consistency checking has empowered business users in ad hoc situations by providing the necessary support, allowing them to focus on and engage actively in their work instead of being locked out due to the rigidness of process definitions.

To summarize, business users have been empowered to react ad hoc on individually assessed situations within the guardrails defined by compliance rules. This was achieved by the combination of: (i) predefined processes whose behavioral and structural consistency is checked by model checking techniques during design time, and (ii) flexibility, which is supported by on-the-fly checking during case execution of ad hoc added goals and tasks. Business users select ad hoc actions from a set of templates. When being added to the case, tasks were checked for overall behavioral consistency. Business users acknowledge the advantage of working goal-oriented allowing them to define the actually needed tasks and their execution sequences as they need within the guardrails of compliance rules defined in a business specific language. Our decision in introducing the aforementioned domain-specific constraint language supporting the business users' domain of expertise was well accepted. As the domain-specific ontology concepts can map the elements of the compliance rules with the underlying data models, the administrators were able to better relate high-level compliance requirements with technical elements such as cases, process fragments, data objects, etc. The possibility to define voting tasks with freely configurable check list items without IT involvement for data model extensions is also seen as an important factor for enabling flexibility in an agile business environment.

Compliance rules are centrally managed and associated with specific cases they shall apply to. The ACM stakeholders also perceived positively the categories of consistency rules. That is, two types of business rules have provided a responsive solution for ad hoc changes in ACM: stated-based and data-based. State-based rules are used by the business administrator or an empowered business clerk to define compliance rules without the support from IT staff. Data-based rules are applied when the available data models allow explicit access to data attributes in order to constrain the execution of goals and/or tasks of a case.

Our study is one of the first attempts in applying existing formal modeling and verification techniques to the domain of ACM and in a particular ACM system. Thus, there is room for future work. For instance, to address the organization of compliance rules and how to deal with compensation of acceptable compliance rule violations.

Also how ontologies can be effectively applied to formulate domain specific business rules by business users (non-IT) needs further investigations. We also aim to expand our findings to other ACM applications and emerging standards. This will help to improve and strengthen our findings in a broader context, and underline the application of formal methods in highly flexible and knowledge intensive systems.

Acknowledgement. This work was supported by the FFG project CACAO, no. 843461 and the Wiener Wissenschafts-, Forschungs- und Technologiefonds (WWTF), Grant No. ICT12-001.

References

1. van der Aalst, W.M.P., Weske, M., Grünbauer, D.: Case handling: a new paradigm for business process support. Data Knowl. Eng. **53**(2), 129-162 (2005)
2. Reichert, M., Weber, B.: Enabling Flexibility in Process-Aware Information Systems: Challenges, Methods, Technologies. Springer, Heidelberg (2012)
3. Ly, L.T., Rinderle, S., Dadam, P.: Integration and verification of semantic constraints in adaptive process management systems. Data Knowl. Eng. **64**(1), 3-23 (2008)
4. Fellmann, M., Zasada, A.: State-of-the-art of business process compliance approaches: a survey. In: 22nd European Conference on Information Systems. Tel Aviv, Israel (2014)
5. Dwyer, M.B., Avrunin, G.S., Corbett, J.C.: Patterns in property specifications for finite-state verification. In: Proceedings of the 21st international conference on Software engineering. New York (1999)
6. ISIS Papyrus: Adaptive Case Management. `http://www.isis-papyrus.com/e15/pages/business-apps/acm.html`
7. Liu, Y., Müller, S., Xu, K.: A static compliance-checking framework for business process models. IBM Syst. J. **46**(2), 335-361 (2007)
8. Namiri, K., Stojanovic, N.: Pattern-Based Design and Validation of Business Process Compliance. In: Meersman, R., Tari, Z. (eds.) On the Move to Meaningful Internet Systems 2007: CoopIS, DOA, ODBASE, GADA, and IS, pp. 59-76. Springer, Berlin (2007)
9. Awad, A., Decker, G., Weske, M.: Efficient Compliance Checking Using BPMN-Q and Temporal Logic. In: Dumas, M., Reichert, M., Shan, M. (eds.) Business Process Management: 6th International Conference, BPM 2008, Milan, Italy, September 2-4, 2008, Proceedings, pp. 326-341. Springer, Berlin (2008)
10. van der Aalst, W.M.P., de Beer, H.T., van Dongen, B.F.: Process Mining and Verification of Properties: An Approach Based on Temporal Logic. In: Meersman, R., Tari, Z. (eds.) On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE, pp.130-147. Springer, Berlin (2005)
11. Birukou, A., D'Andrea, V., Leymann, F., Serafinski, J., Silveira, P., Strauch, S.,

Tluczek, M.: An Integrated Solution for Runtime Compliance Governance in SOA. In: Maglio, P.P., Weske, M., Yang, J., Fantinato, M. (eds.) Service-Oriented Computing: 8th International Conference, ICSOC 2010, San Francisco, CA, USA, December 7-10, 2010, Proceedings, pp. 122-136. Springer, Berlin (2010)

12. Holmes, T., Mulo, E., Zdun, U., Dustdar, S.: Model-aware Monitoring of SOAs for Compliance. In: Dustdar, S., Li, F. (eds.) Service Engineering, pp. 117-136. Springer, Berlin (2011)
13. Ly, L.T., Maggi, F.M., Montali, M., Rinderle-Ma, S., van der Aalst, W.M.P.: A Framework for the Systematic Comparison and Evaluation of Compliance Monitoring Approaches. In: Enterprise Distributed Object Computing Conference (EDOC), IEEE 17th International, pp. 7-16 (2013)
14. Swenson, K.: Case Management: Contrasting Production vs. Adaptive. In: Fischer, L. (ed.) How Knowledge Workers Get Things Done, pp. 109-116. Future Strategies Inc., Lighthouse Point, FL (2012)
15. Tran, T.T.K., Pucher, M.J., Mendling, J., Ruhsam, C.: Setup and Maintenance Factors of ACM Systems. In: Demey, Y.T., Panetto, H. (eds.) On the Move to Meaningful Internet Systems: OTM 2013 Workshops, pp. 172-177. Springer, Berlin (2013)
16. Espertech: Complex Event Processing (CEP). http://www.espertech.com
17. Governatori, G., Rotolo, A.: Norm Compliance in Business Process Modeling. In: Dean, M., Hall, J., Rotolo, A., Tabet, S. (eds.) Semantic Web Rules - International Symposium, RuleML 2010, Washington, DC, USA, October 21-23, 2010. Proceedings, pp. 194-209. Springer, Berlin (2010)
18. Sem, H.F., Carlsen, S., Coll, G.J.: Combining Compliance with Flexibility. In: Fischer, L. (ed.) Thriving on Adaptability: Best Practices for Knowledge Workers, pp. 59-71. Future Strategies Inc., Lighthouse Point, FL (2015)
19. Tran, T.T.K., Ruhsam, C., Pucher, M.J., Kobler, M., Mendling, J.: Towards a Pattern Recognition Approach for Transferring Knowledge in ACM. In: Enterprise Distributed Object Computing Conference Workshops and Demonstrations (EDOCW), IEEE 18th International, pp. 134-138 (2014)
20. van der Aalst, W.M.P., Pesic, M.: DecSerFlow: Towards a Truly Declarative Service Flow Language. In: Bravetti, M., Núñez, M., Zavattaro, G. (eds.) Web Services and Formal Methods, 3rd International Workshop, WS-FM 2006 Vienna, Austria, September 8-9, 2006, Proceedings, pp. 1-23. Springer, Berlin (2006)
21. United States Environmental Protection Agency: Lead Renovation, Repair and Painting Program Rules. http://www2.epa.gov/lead/lead-renovation-repair-and-painting-program-rules