Enabling Flexibility of Business Processes Using Compliance Rules: The Case of Mobiliar

Thanh Tran Thi Kim, Erhard Weiss, Christoph Ruhsam, Christoph Czepa, Huy Tran, and Uwe Zdun

Abstract

(a) **Situation faced**: Insurance case work can follow established procedures only to a certain degree, as the work depends upon experienced knowledge workers who decide the best solutions for their clients. To produce quality documents in such a knowledge-intensive environment, business users of Die Mobiliar, the oldest private insurance company in Switzerland, were guided by a wizard application that enabled them to compose insurance documents from predefined building blocks in a series of pre-defined steps. As these steps were hardcoded into the wizard application, the processes could not adapt quickly enough to accommodate new insurance products and associated documentation. Rapidly changing insurance markets produce new types of documents daily, so business users must react flexibly to client requests. Although fully automated processes can be defined when sufficient process knowledge exists, they seriously hinder the innovation and business agility that is critical in insurance markets.

(b) **Action taken**: To overcome this problem, Die Mobiliar uses the Papyrus Communication and Process Platform (http://www.isis-papyrus.com/e15/pages/software/platform-concept.html) as the basis for its customized "Mobiliar Korrespondenz System" (MKS, Mobiliar Correspondence System), with full functionality for online interactive business document production (ISIS Papyrus). Our approach combines automatically executed

T.T.T. Kim • E. Weiss • C. Ruhsam (✉)
ISIS Papyrus Europe AG, Maria Enzersdorf, Austria
e-mail: thanh.tran@isis-papyrus.com; erhard.weiss@isis-papyrus.com; christoph.ruhsam@isis-papyrus.com

C. Czepa • H. Tran • U. Zdun
Research Group Software Architecture, University of Vienna, Vienna, Austria
e-mail: christoph.czepa@univie.ac.at; huy.tran@univie.ac.at; uwe.zdun@univie.ac.at

J. vom Brocke, J. Mendling (eds.), *Business Process Management Cases*, Management for Professionals, DOI 10.1007/978-3-319-58307-5_6

business compliance rules with process redesign to provide the flexibility that is essential for insurance processes. The original processes are split into reusable sub-processes, accompanied by a set of ad hoc tasks that the business users can activate at runtime to meet clients' emergent requirements. A set of compliance rules guarantees that the process conforms to corporate and regulatory standards.

(c) **Results achieved**: The business compliance rule approach has two primary benefits: (i) company management has a process that is well-documented and provably compliant, and (ii) the business users can respond flexibly to their clients' needs within the boundaries of defined compliance rules, thus improving the customer experience. The flexibility achieved by this approach allows business users to adapt their insurance processes, an advantage from which the whole insurance industry can benefit. The redesigned process with few reusable core elements, combination with a set of ad hoc tasks, decreases the number of process templates (wizard processes) that are required to handle unpredictable situations. A smaller template library also reduces maintenance efforts for business administrators.

(d) **Lessons learned**: Rigid process modeling is not suitable for highly dynamic business domains, like the insurance industry, that are moving into the digital era. Instead, a hybrid of declarative and imperative modeling is best suited to such domains. Our approach provides a maximum of flexibility within mandated constraints, enabling businesses to adapt to changing market requirements with minimal involvement by IT departments. In order to set expectations properly, the use of the two modeling types should be transparent to business users. The adoption of the new approach happens gradually to cope with business considerations like the integration of compliance checking into Die Mobiliar's production system.

1 Introduction

The Swiss insurance company Die Mobiliar is the oldest private insurance organization in Switzerland. As a multiline insurer that offers a full range of insurance and pension products and services, Die Mobiliar handles a large number of documents, which are exchanged with approximately 1.7 million customers (Mobiliar: Die Mobiliar Versicherungen und Vorsorge). An insurance document issued by Die Mobiliar is not only a piece of paper; it serves as a business card, representing the company to its customers. Moreover, Die Mobiliar considers well-designed documents that are rich in content as an opportunity to communicate and build a strong relationship with its customers.

Die Mobiliar uses the Papyrus Communication and Process Platform[1] as the basis for its customized "Mobiliar Korrespondenz System" (MKS), with full

[1] http://www.isis-papyrus.com/e15/pages/software/platform-concept.html

functionality for online interactive business document production (ISIS Papyrus). MKS uses wizard processes to assist several thousand business users in handling various types of documents related to insurance cases. A wizard contains steps that guide business users through the document-generation process. Therefore, the wizard resembles a dedicated imperative process modelled in BPMN (BPMN Specification) for guiding users through a sequence of activities. To be effective in generating high-quality documents, these processes must be well-prepared and suitable for such a large and complex work environment. They must also comply with requirements involving laws, contracts with business partners, general standards, best practices, and civil and corporate regulations. A rigid process configuration ensures that the process execution remains controlled and satisfies these compliance requirements; however, it does not consider the workers' tacit business knowledge, which is usually an underestimated source of compliance with regulations (Governatori and Rotolo 2010). To be able to react quickly to changing business needs, modern business systems must be under the control of business departments that depend only minimally on IT departments. Process changes and introductions of new applications must be accomplished in days or weeks, not months or years. Rigid wizard processes that are predefined in the design process and executed sequentially at runtime cannot meet such modern requirements. The rapidly changing insurance markets generate requirements for new types of documents daily, obliging business users to react quickly to client requests. A case that requires insurance documents to be generated can follow established procedures only to a certain extent, as the task usually depends on knowledge workers' finding the best solutions for their clients. In our insurance context, knowledge work is performed by insurance clerks, thus we will use the term clerks to represent knowledge workers through our paper. This purpose is what Adaptive Case Management (ACM) (Tran Thi Kim et al. 2013) is designed for: customer-oriented work driven by goals contained in a case that allows the clerks to choose the appropriate actions from a context-sensitive set of ad hoc tasks with needed data and content to fulfill the related goal.

In this paper, we show how to simplify the wizard design and enhance the flexibility of its execution using a compliance-rule-and-consistency-checking system embedded in an ACM system (Tran Thi Kim et al. 2015). Instead of mapping the entire process into predefined task sequences, the system offers a selection of up-coming tasks at runtime, which are governed by compliance rules defined by business administrators at design time. Clerk must adhere to the loosely interrelated task sequences defined by these rules but can decide which tasks will be executed based on the workers' ad hoc assessment of the situation. Thus, a case evolves gradually instead of being predefined by business administrators who cannot predict all knowledge-intensive scenarios.

As a consequence, this approach guides both business users at runtime when they select from ad hoc task templates and business administrators at design time when they define new or amend existing sub-processes. The consistency-checking system consists of a model checker (Czepa et al. 2015) that supports process administrators at design time, and an on-the-fly compliance checker (Czepa et al.

2016a) that observes clerks' behavior at runtime to ensure that both activities comply with company regulations. We restructure the wizard process and combine it with compliance rules in order to provide optimal flexibility for business users during process execution. The resulting approach can be applied to any knowledge-intensive domain and is not specific to the insurance industry.

2 Situation Faced

MKS is built on the ACM and Correspondence Solution of the Papyrus platform. While the Correspondence Solution handles the design and content of documents, the ACM solution's process modeling and execution capabilities manage the processes involved in document generation.

MKS enables clerks to generate documents interactively based on wizard process templates and to retrieve data dynamically from various business systems. The wizard is an ACM case that defines processes comprised of interactive user steps to be executed by the clerks, as well as service tasks, such as web services, that the system executes automatically for data retrieval. A document template that is composed of text-building block templates with embedded data, variables, and/or logic is used as artifact for steps in the wizard process. Forms, as integral parts of the wizard definition, request that the clerks enter the document data as variable values, populating the document in a step-by-step approach. Figure 1[2] shows a typical wizard input form and the preview of the corresponding document at a certain stage of the document generation process. The entered data are imported directly to the document.

As shown in Fig. 2, the wizard steps executed by clerks at runtime are prepared by business administrators as templates and stored in a template library at design time. The processes are defined with an editor that has full functionality to edit, visualize, and simulate the execution of wizards before they are released into production. Transitions connect the steps, and each step defines actions that select and add text building blocks to the document.

The motivation for this paper is to support unpredictable or unlikely situations that could explode the number of process variants. Suppose that a clerk recognizes during wizard execution that the customer's address is incorrect and must be updated in the external system that was queried by the service task. In this case, the clerk should be able to edit the address right in the wizard form and notify the data's owner about the change. The clerk, who is engaged in a strict wizard process, would benefit from the ability to perform ad hoc actions in this situation. In order to support flexibility at runtime, we address this challenge by applying consistency-checking methods in combination with compliance rules, as discussed in the next section.

[2]This figure and others show original screenshots from Mobiliar's business application. Relevant items in German are translated as needed.

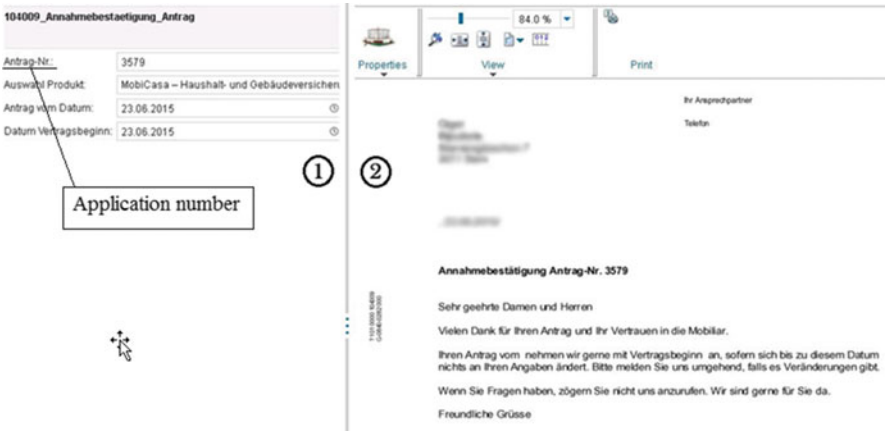

Fig. 1 Wizard step with data form (1) and document preview (2). The value "3579" for application number is entered in the form on the *left side* and simultaneously displayed in the document preview of the related building block on the *right*

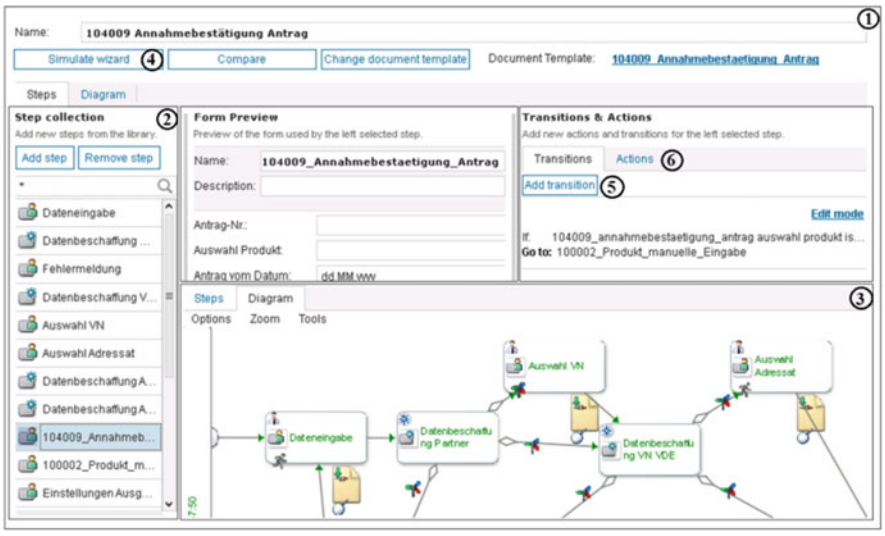

Fig. 2 Wizard template composition editor (1) with functionality to edit (2), visualize (3) and simulate (4) the execution of a wizard containing steps that are connected by transitions (5) and defined with actions (6)

3 Action Taken

Our approach introduces a generically applicable consistency-checking method to enable a more flexible execution of document-creation wizards. The following subsections outline the approach, describe our extension of the wizards with ad hoc tasks, and explain how to guarantee these flexible processes' compliance through a set of managed compliance rules. In the original operating principle of MKS, shown in Fig. 3a, business administrators defined a wizard process template at design time, which was instantiated by clerks for execution at runtime. Clerks strictly followed the steps predefined in the wizard to create a document, but because the system did not allow clerks to adapt the process at runtime, they could not react to unforeseen situations within an insurance case.

The overview of MKS extended by the consistency-checking system is provided in Fig. 3b. In this approach, a wizard ACM case template (Tran Thi Kim et al. 2013) is assembled at design time from goals that are achieved through predefined sub-processes and/or individual ad hoc tasks. Each sub-process is attached to the goal and combines the necessary tasks in a particular sequence. The quality of case templates is ensured by means of model-checking (Czepa et al. 2015) before the templates are released. At runtime, the set of compliance rules assigned to the case checks the execution of case elements—goal instances, process instances, task instances, and ad hoc actions—on the fly (Czepa et al. 2016a). Clerks can follow the steps defined in the templates while performing ad hoc actions to adapt to new situations. A goal is reached when all its tasks are completed and the attendant data

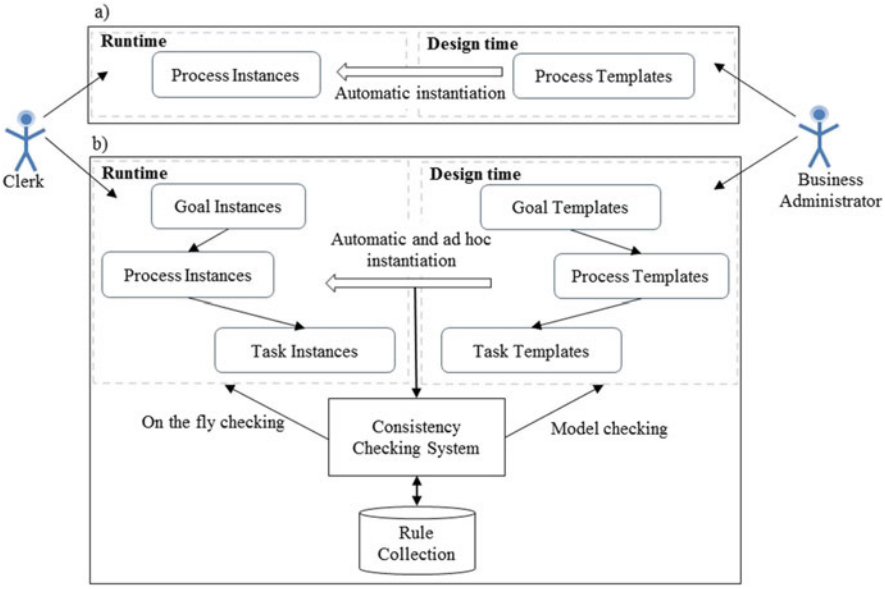

Fig. 3 (**a**) MKS operating principle and (**b**) MKS with consistency checking

are acquired. Consequently, a document is finished when all goals of a wizard are reached.

3.1 Design of Compliance-Rules-Enabled Wizards

Figure 4a shows the original MKS wizard process, which can be divided into beginning, middle, and end parts of the process. The beginning part of the process retrieves the client's personal data and selects an insurance product. The end part defines the document-delivery channel, while the middle part is comprised of the steps that are necessary for the specific insurance case. The system's analysis of the original wizard processes led to the simplified model in Fig. 4b. Numerous processes share the same beginning and ending parts, but the middle part of each process is distinct from the others, although they might have tasks in common.

In our approach, the original wizard processes are transformed into flexible processes with a goal-driven structure. The beginning and ending parts are modeled as predefined sub-processes that can be reused in various cases. The middle part is split into several individual ad hoc tasks. A case that uses the restructured processes

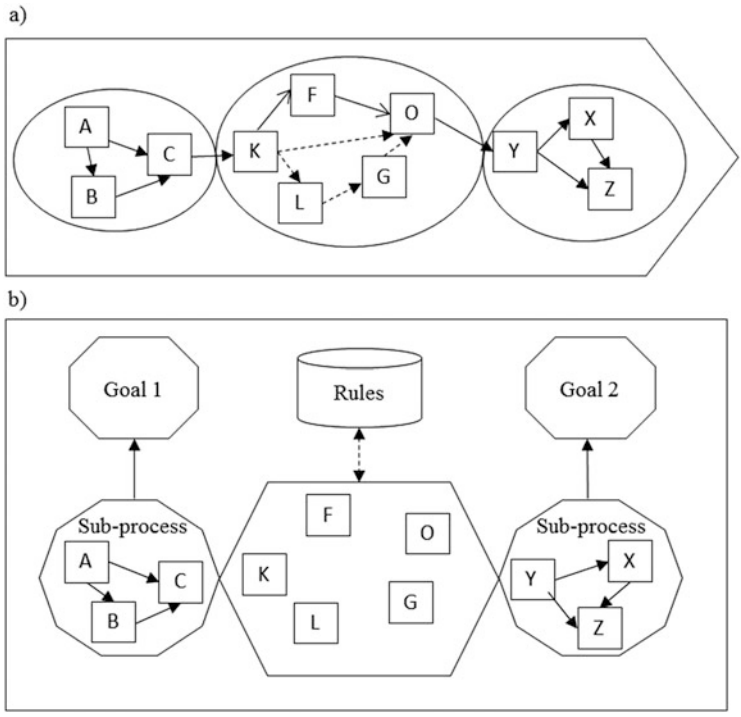

Fig. 4 (**a**) Typical MKS processes and (**b**) Flexible MKS processes. The tasks in this figure are anonymized as *squares* with capital letters and grouped. *Arrows* indicate the direction of process execution, and *dashed lines* represent alternative workflows of the middle part

is driven by two goals: *Goal 1*, defining the beginning sub-process, and *Goal 2*, defining the ending sub-process. The tasks of the middle part are either related to *Goal 1* or *Goal 2* or could be assigned by the clerks to newly defined goals. If the tasks of the middle part must follow a certain execution sequence, they are monitored by associated compliance rules. In other words, the rules define the boundaries of the middle part without rigidly predefining the processes. The consistency-checking system enables clerks to maintain compliance by highlighting tasks that do not comply with one or more rules. If certain tasks are optional and do not have to follow any specific sequence, the business users can add them as needed. Because the same rules are active when business administrators define process templates, the rules enforce compliance at design time as well.

3.2 Constraint Definitions Using Compliance Rules

In order to govern the middle part of a wizard process, we use state-based and data-based rules. State-based rules define the sequences of tasks based on their states, such as "started," "finished," and "running." For example, a sequence from Task K to Task F can be described by the rule, "Task F can be started only after Task K is finished." This rule is expressed in the constraint language as.

```
Constraint No1 for MobiliarCase{
K.finished leads to F.started }
```

In this example the states of tasks *K* and *F* are *finished* and *started*, respectively. The temporal pattern of type precedence is defined by the keywords *leads to*.

To define the temporal patterns in our system, we adapt temporal expressions from the patterns defined by Dwyer et al. (1999):

- Existence: K.finished *occurs*
- Absence: K.finished *never occurs*
- Response: K.finished *leads to* F.started. In other words, only if K.finished happens, can F.started happen.
- Precedence: K.finished *precedes* F.started. In other words, F.started can happen only if K.finished has happened.

Data-based rules enable business users to define task dependencies that are related to data conditions. State-based and data-based rules can be combined to express a compliance requirement. For example, Task F can be started only when Task K is finished and the value of a certain data attribute meets a certain requirement, such as the customer's birth year is greater than or equal to 1981.

```
Constraint No2 for MobiliarCase{
(K.finished and CustomerBirthyear >= 1981) leads to F.started
}
```

In unforeseen circumstances, the underlying data models might not provide access to critical data. In order to support flexibility in such situations without the need for explicit data definitions by IT, business users can check conditions manually using voting tasks that are guarded by compliance rules. Let us assume a voting task called "*Inquire additional customer interests*" must be concluded before the final pricing can be finished:

```
Constraint No3 for MobiliarCase{
InquireInterest.approved leads to Pricing.finished
}
```

Voting tasks like *InquireInterest* can be employed quickly to adapt to new situations. The business administrator can create the task template without the support of database experts or IT people and can specify with checklists which items must be verified with the customer. Alternatively, the business user can define the checklist at runtime to adapt even more dynamically to the current situation. Unstructured data is popular in real-life systems since data definitions cannot be amended quickly in IT systems with bureaucratic change-management cycles. In a car insurance case, for example, the result of an investigation into whether the car was damaged intentionally or accidentally can be reported by means of a simple voting task decided by a clerk.

Since MKS is based on the ISIS Papyrus ACM framework, processes and tasks are reusable components of the ACM system to be shared with other goals and cases. The sequence of the tasks in the sub-processes is modeled with transitions and gateways following BPMN standards (BPMN). The tasks of the middle part are ad hoc tasks selected by the clerks at runtime. Each of these tasks can be added to the case when the clerk sees the need to do so based on the case's content or context. The tasks' order of execution is not predetermined but is constrained by rules. A User Trained Agent (UTA) implemented in the Papyrus ACM system further assists the clerk in new situations by suggesting best next actions that were learned earlier from similar situations faced by other users (Tran et al. 2014). Thus, knowledge acquisition and sharing becomes an integral part of the business application, enabled by the business intelligence component, UTA.

To demonstrate the results achieved in applying our approach, we used the original wizard case *Acknowledgement of Application* (Fig. 5). Before the redesign, an ACM wizard case was completely driven by a predefined process that contained all of the steps of the wizard. In this insurance case, a clerk created a document that confirmed the successful submission of an insurance application. First, the clerk entered some identification numbers, such as the insurance ID, customer ID, or case ID, into the system. The customer's data was retrieved by the predefined process through web service tasks from various sources based on the entered data. Then the clerk selected the matching insurance holder and address and inserted specific information for the particular insurance case. A document confirming the

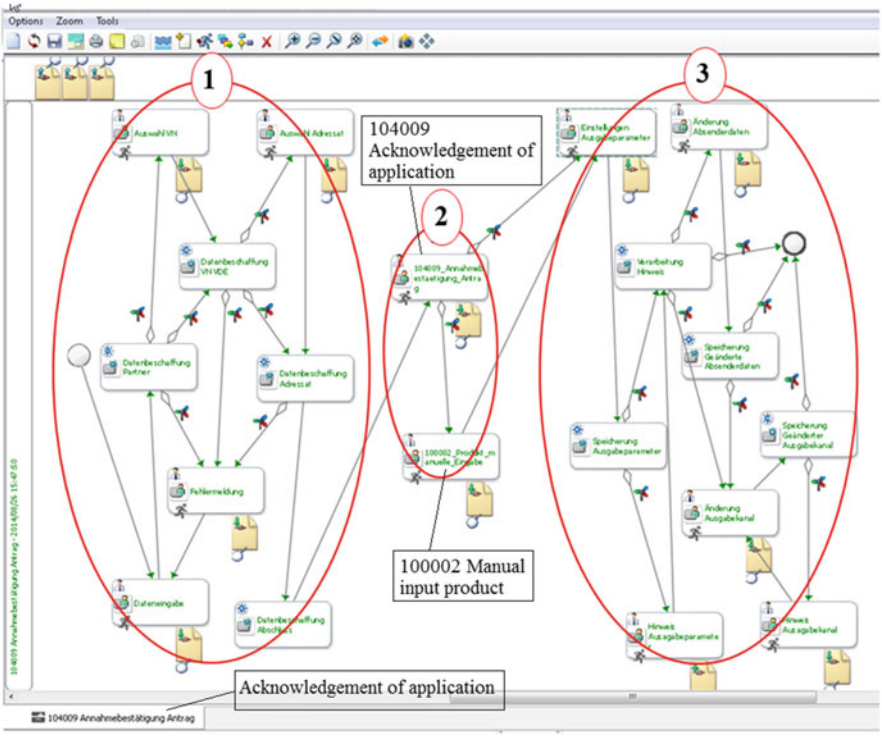

Fig. 5 Process model before redesign analyzed into three parts

acceptance of the insurance application was generated and sent to the customer based on the output channel determined by the clerk at the end of the process.

An ACM wizard case is not driven by predetermined steps, but by goals that are fulfilled by the clerk. The redesigned wizard process of the *Acknowledgement of Application* case is divided into three parts, as illustrated in Fig. 4. The first and last parts require no flexibility and are linked to the goals of predefined processes. The clerk can freely add the flexible part's tasks at runtime, and their sequence is determined, if necessary, by the compliance rules introduced in our approach.

Like any other ACM case, the redesigned *Acknowledgement of Application* ACM case has associated goals, processes, and tasks. The *First Core Goal* holds the beginning part of the process, which is configured as a sub-process for retrieving insurance customer data from the database. The *Last Core Goal* contains the ending part of the wizard process for choosing the channel by which the document will be delivered to the customer. The tasks of the middle part are not predefined in the wizard template but are added by the business user as necessary at runtime to address the specific customer situation. In this specific case, the ad hoc task templates are prepared as *manual input product* and *acknowledgement of application*.

The task execution of the middle part is controlled by three compliance rules, defined by business administrators. For example, rule *R0* may express that the task *acknowledgement of application* must be present at least one time, while *R1* defines the dependency of task *acknowledgement of application* on task *manual input product* when the *selection product* is *manual input*, and Rule *R2* expresses the dependency of task *acknowledgement of application* on task *selection output channel* when the *selection product* is not *manual input*.

```
Constraint R0 for MobiliarCase{
acknowledgement_of_application.started occurs at least 1x
}
Constraint R1 for MobiliarCase{
  (acknowledgement_of_application.finished  and  selection_product
equal to "manual_input") leads to manual_input_product.started
}
Constraint R2 for MobiliarCase{
 (acknowledgment_of_application.finished and selection_product not
equal to "manual_input") leads to selection_output_channel.started
}
```

The two data-based rules *R1* and *R2* can be visually simplified by an alternative expression using a voting task to check whether the *selection product* is *manualinput*. The voting task is named *check_selection_product_manual_input*.

```
Constraint R3 for MobiliarCase{
      (acknowledgment_of_application.finished      leads      to
check_selection_product_manual_input.started
}
Constraint R4 for MobiliarCase{
    check_selection_product_manual_input.approved     leads     to
manual_input_product.started
}
Constraint R5 for MobiliarCase{
     check_selection_product_manual_input.denied     leads     to
selection_output_channel.started
}
```

Compliance rules are composed in natural language by business administrators using the Papyrus rule editor, as shown in Fig. 6. To facilitate that activity, the editor offers a selection of elements from a list of items using business terminology. Thus, the language of business is used to define the rule with auto-completion features as the user types.

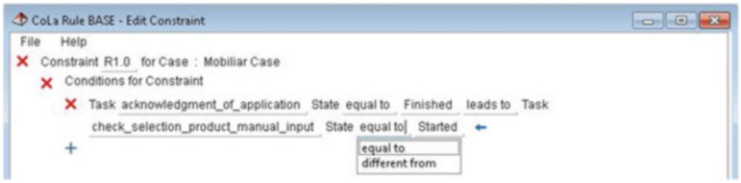

Fig. 6 Compliance rule editor

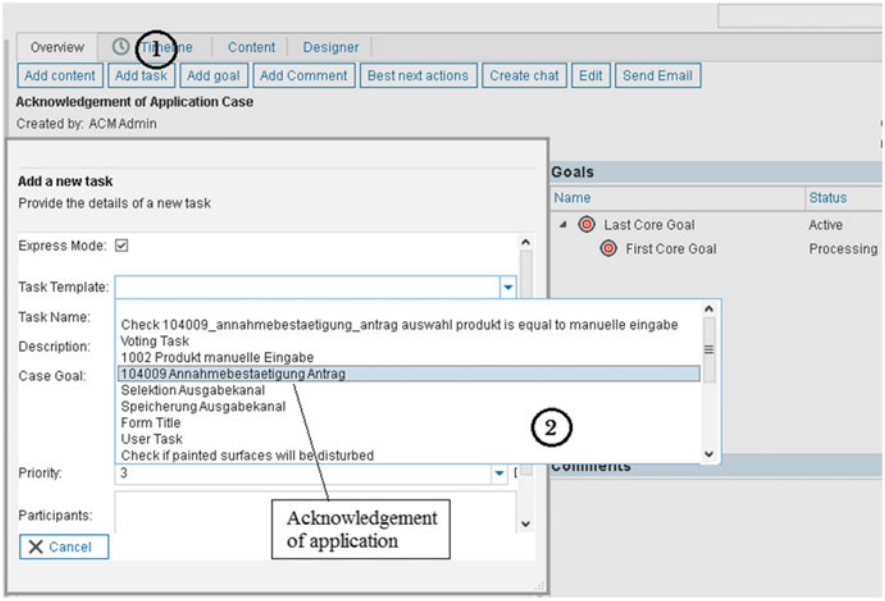

Fig. 7 Add an ad hoc task at runtime by the function Add task (1) with a list of task templates (2)

At runtime, when a clerk creates a confirmation for a customer who has submitted an application, an instance of the *acknowledgement of application* case is created upon the clerk's selection of that template.

During processing of the *First Core Goal*, the clerk is presented with a form to enter the insurance ID, customer ID, or case ID. When all of the first core goal's predefined tasks are finished, the clerk adds task *acknowledgement of application*, as suggested by the consistency-checking system, which is used to create a confirmation of an application. Figure 7 demonstrates the use of function *Add task* with a dialog showing a list of task templates that are provided to the clerk for adding an ad hoc task on the fly.

Although the task *acknowledgement of application* is suggested to the clerk because Constraint *R0* was temporarily violated, the clerk can do other tasks as well. However, as soon as an ad hoc task related to the compliance rule constraint is

added, it will be controlled by the consistency-checking system. The constraint that defines the occurrence of tasks is used to ensure the presence of the task that initiates the variable middle parts, like the task *acknowledgement of application*. Therefore, to complete a case successfully, the clerk must eventually execute that task.

When task *acknowledgement of application* is finished, the constraints *R1* and *R2* are investigated by the consistency-checking system. Since task *acknowledgement of application* is finished, and if the *selection product* is chosen as *manual input*, the consistency-checking system suggests the task *manual input product* to the clerk. If *selection product* is not chosen as *manual input,* task *selection output channel* is suggested to the clerk. Thus, the execution of the ad hoc tasks is controlled by the consistency-checking system so the clerk does not overlook any tasks that would violate the case's compliance with the rules. The user can also consult the experience of other users who have been in the same situation by asking the UTA for best next actions. When there are no more suggestions, the clerk can continue the steps defined in the *Last Core Goal*. When the goal is reached, a confirmation document for the application is generated and the case is closed.

In summary, compliance rules can control ad hoc tasks added at runtime when business compliance requirements demand it. Some of these tasks must be executed in a certain order and/or depend on the availability of certain data, which is defined by a set of compliance rules. Other tasks that do not require such control can be added at the clerk's discretion, such as when the clerk institutes an *add additional information* task when the document is lacking required information. Therefore, our approach enables clerks to add ad hoc tasks—tasks that may not have been foreseen when the wizard was initially designed—at runtime under the control of the compliance rule system based on the current context.

4 Results Achieved

The ACM technology used to build the wizard processes supports the definition of tasks to be performed by business staff at design time and their selective application by knowledge workers at runtime so Die Mobiliar can react quickly to new business requirements without involving IT. Instead of defining rigid process models that IT must implement with lengthy change-management and rollout cycles, the processes can be defined directly by Die Mobiliar's business administrators using a process editor built on the Papyrus platform.

MKS's ability to edit wizard templates at any time enables Die Mobiliar to define new document and wizard templates within the boundaries imposed by the predefined processes. The process management in ACM is highly flexible, as it supports both automatic and ad hoc actions (Tran Thi Kim et al. 2013). Although fully automated processes can be defined for well-behaved and predictable domains, they hinder the innovation and business agility that is critical in insurance markets. The clerks who come face-to-face with insurance situations should have the flexibility to adapt the case at runtime.

The enhanced structure of the ACM wizard gives clerks immediate flexibility while staying within the boundaries imposed by compliance rules. Clerks can institute goal and task templates manually using the predefined wizard case for adding new actions at runtime. A set of compliance rules in a constraint-specification language examines the consistency of the tasks performed and verifies process compliance (Czepa et al. 2016b).

Compliance rules are also enforced at design time through model-checking (Clarke 2008; Czepa et al. 2015, 2016a) when business administrators develop sub-process templates. Model-checking verifies the structural consistency of the predefined sub-processes. By observing compliance rules at design time as well as at runtime, business administrators and clerks are prevented from violating compliance requirements, and dynamically assembled sequences of tasks are guaranteed to meet the same structural criteria that are applied to predefined wizard processes. Thus, the boundaries defined by the rule system ensure the compliance of the overall case execution. This approach confers a significant benefit during the change-management and release process by reducing tests and error-correction efforts.

With MKS's redesigned structure, the goal, process, and task templates can be reused in various wizard cases, so the number of predefined process templates in the library can be reduced considerably. Based on the subset of wizard process templates from Die Mobiliar that we could use for our study, we estimated a 40–90% reduction, depending on the degree of the core process templates' standardization and their efficient reuse. To that end, goals and related sub-processes that appear in several wizards can be predefined in the wizard case at design time. The reuse of shared items will improve the quality and consistency of related cases and avoid redundancies, which are always a source of inconsistency, especially in large-scale and continuously evolving systems. Clerks can process the variable tasks between the predefined processes at runtime, and ad hoc tasks instituted from task templates can be added to adapt to unforeseen situations that require new documents. By defining a case partially at design time and completing it with variable and ad hoc tasks at runtime, Die Mobiliar can avoid inordinately complex wizard cases.

5 Lessons Learned

The trade-off between comprehensibility and flexibility in business process modeling has been addressed by both academia and industry (De Smedt et al. 2016), and declarative and imperative models have been studied to improve the flexibility of process models (Fahland et al. 2009, 2010; Haisjackl et al. 2016; Prescher et al. 2014). To address this challenge, we introduced a theoretical approach and its successful application in the hybrid declarative-imperative modeling and enactment of a business process. We learned lessons from this practical application and case study.

A rapidly changing industry like insurance presents a plethora of unpredictable business situations. By attempting to cover all business cases up front, rigid process modeling of such markets produces bloated process template libraries that hinder an organization's ability to respond to emergent requirements. The construction and maintenance of such systems consume significant effort and resources (ISIS Papyrus).

No specific discovery methodology was applied in this case study; the discovery for the process redesign was based on the designers' experience. The shared portions of hundreds of process templates were discovered and manually extracted as sub-processes. In the resulting approach, the tasks in the variable, transitional part of the process—that is, the middle part—are loosely connected by constraints. Depending on the designers, the relationships between two tasks are defined either declaratively by constraints or imperatively in a sub-process. The hybrid model can be evolved gradually through multiple iterations to improve the enterprise's productive system.

The inherent flexibility of declarative models makes them suitable to the goal-oriented approach of ACM in providing an adaptive and flexible system to deal with unpredictable business events. Our case study employs an ACM framework that supports the application of imperative models to reusable sub-processes and of declarative models to ad hoc actions within a case structure. The duality of modeling is hidden from business users, as the associated steps can be instituted automatically, making case execution transparent.

The conversion between imperative and declarative models has been addressed by various studies (De Smedt et al. 2016; Prescher et al. 2014), which focus on how to obtain a set of declarative constraints from an imperative model or vice versa, or even how to combine them into a hybrid model. Our case study is unique in that regard, as it does not consolidate the two model types into a hybrid model at design time but incorporates them separately into the process instances, which are manipulated by knowledge workers at runtime without explicit modeling. The compliance-checking employed by our approach ensures the consistency of the execution by suggesting activities and preventing user mistakes within the boundaries described by the applied compliance rules. We kill two birds with one stone: compliance rules maintain compliance automatically, and they provide business users the freedom to decide which tasks will best achieve their business goals based on their own experience.

Lessons were also learned from a practical perspective. Die Mobiliar appreciated the benefits of this approach, which enables a customer-oriented business strategy that focuses on service quality and the customer's experience. In the midterm Die Mobiliar will look into changing several of its predefined process models into flexibly managed workflows. However, such a change would involve a paradigm change, and its adoption will occur gradually, as the company must also address considerations like the installation of new business user responsibilities for the integration and maintenance of the consistency-checking solution in the production system.

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

References

BPMN. *Specification—Business process model and notation*. Accessed July 14, 2016, from http://www.bpmn.org/

Clarke, E. M. (2008). The birth of model checking. In O. Grumberg & H. Veith (Eds.), *25 years of model checking: History, achievements, perspectives* (pp. 1–26). Berlin: Springer.

Czepa, C., Tran, H., Zdun, U., Rinderle-Ma, S., Tran Thi Kim, T., Weiss, E., & Ruhsam, C. (2015). Supporting structural consistency checking in adaptive case management. In *International Conference on Cooperative Information Systems (CoopIS)* (pp. 311–319).

Czepa, C., Tran, H., Zdun, U., Tran Thi Kim, T., Weiss, E., & Ruhsam, C. (2016a). Towards a compliance support framework for adaptive case management. In *5th International Workshop on Adaptive Case Management and other Non-workflow Approaches to BPM (AdaptiveCM 16), 20th IEEE International Enterprise Computing Workshops (EDOCW 2016)*.

Czepa, C., Tran, H., Zdun, U., Tran Thi Kim, T., Weiss, E., & Ruhsam, C. (2016b). Ontology-based behavioral constraint authoring. In *2nd International Workshop on Compliance, Evolution and Security in intra- and Cross-Organizational Processes (CeSCoP 2016), 20th IEEE International Enterprise Computing Workshops (EDOCW 2016)*.

De Smedt, J., De Weerdt, J., Vanthienen, J., & Poels, G. (2016). Mixed-paradigm process modeling with intertwined state spaces. *Business Information System Engineering, 58*, 19–29.

Dwyer, M. B., Avrunin, G. S., & Corbett, J. C. (1999). Patterns in property specifications for finite-state verification. In *Proceedings of the 21st International Conference on Software Engineering* (pp. 411–420). New York: ACM.

Fahland, D., Lübke, D., Mendling, J., Reijers, H., Weber, B., Weidlich, M., & Zugal, S. (2009). Declarative versus imperative process modeling languages: The issue of understandability. In T. Halpin, J. Krogstie, S. Nurcan, E. Proper, R. Schmidt, P. Soffer, & R. Ukor (Eds.), *Enterprise, business-process and information systems modeling: 10th International Workshop*, BPMDS 2009, and *14th International Conference*, EMMSAD 2009, held at CAiSE 2009, Amsterdam, June 8–9, 2009. Proceedings (pp. 353–366). Berlin: Springer.

Fahland, D., Mendling, J., Reijers, H. A., Weber, B., Weidlich, M., & Zugal, S. (2010). Declarative versus imperative process modeling languages: The issue of maintainability. In S. Rinderle-Ma, S. Sadiq, & F. Leymann (Eds.), *Business Process Management Workshops: BPM 2009 International Workshops*, Ulm, September 7, 2009. Revised Papers (pp. 477–488). Berlin: Springer.

Governatori, G., & Rotolo, A. (2010). Norm compliance in business process modeling. In *Semantic Web Rules—International Symposium, RuleML 2010*, Washington, DC, October 21–23, 2010. Proceedings (pp. 194–209).

Haisjackl, C., Barba, I., Zugal, S., Soffer, P., Hadar, I., Reichert, M., Pinggera, J., & Weber, B. (2016). Understanding declare models: Strategies, pitfalls, empirical results. *Software and System Modeling, 15*, 325–352.

ISIS Papyrus. *ISIS Papyrus solution catalog – Swiss Mobiliar*. Accessed March 11, 2016, from http://www.isis-papyrus.com/e15/pages/solutions-catalog/solutions-catalog-mobiliar-wizard.html

ISIS Papyrus. *ISIS Papyrus Press Release*. Accessed August 18, 2016, from https://www.isis-papyrus.com/e15/pages/press/PR20151208-WfMC-Award.html

Mobiliar: Die Mobiliar Versicherungen und Vorsorge. Accessed March, 11, 2016, from https://www.mobi.ch/.

Prescher, J., Di Ciccio, C., & Mendling, J. (2014). From declarative processes to imperative models. In Accorsi, R., Ceravolo, P., & Russo, B. (Eds.), *Proceedings of the 4th International Symposium on Data-driven Process Discovery and Analysis* {(SIMPDA} 2014), Milan, November 19–21, 2014. pp. 162–173. CEUR-WS.org

Tran Thi Kim, T., Pucher, M. J., Mendling, J., & Ruhsam, C. (2013). Setup and maintenance factors of ACM systems. Lecture Notes in Computer Science (including Subser. Lecture Notes in Artificial Intelligence Lecture Notes in Bioinformatics). 8186 LNCS (pp. 172–177).

Tran Thi Kim, T., Weiss, E., Ruhsam, C., Czepa, C., Tran, H., & Zdun, U. (2015). Embracing process compliance and flexibility through behavioral consistency checking in ACM – A Repair Service Management Case. BPM 2015 4th Work. ACM Other Non-Workflow Approaches to BPM (pp. 1–12).

Tran, T., Ruhsam, C., Pucher, M. J., Kobler, M., & Mendling, J. (2014). Towards a pattern recognition approach for transferring knowledge in ACM. In *18th IEEE International Enterprise Distributed Object Computing Conference Workshops and Demonstrations*, EDOCW 2014 (pp. 134–138). doi:10.1109/EDOCW.2014.28.

Thanh Tran Thi Kim is an application researcher working at the Academy Department, ISIS Papyrus Europe AG. She received her doctoral degree in Computer Science at Vienna University of Technology, Austria in December 2012. Her research focuses on adaptive case management, flexibility in process-aware information system, compliance verification. Besides supporting research and development, she participates in cooperation projects between ISIS Papyrus Europe AG and universities. Her main publications are in international conferences, workshops and book chapters for industry cases and best practices of process-aware information systems.

Erhard Weiss is Academy member at ISIS Papyrus Europe AG. He received a bachelor's degree in CIT and a master's degree in Business Process Management and Engineering. His research focuses on Adaptive Case Management.

Christoph Ruhsam is Senior Manager at ISIS Papyrus. He is head of the ISIS Papyrus Academy focusing on applied application research and coordinates scientific projects with external partners. The Academy has a dedicated group for user interface and usability concepts. Another group produces all Papyrus software related documentation and delivers education programs including the ISIS Papyrus Certification. He studied at Vienna University of Technology Communications Engineering and received his doctoral degree in 1994 on digital signal processing of biomedical signals. Since 1995 he works at ISIS Papyrus in a variety of management positions and established in 2003 a dedicated project supervision team (Project-QA), caring for ISIS Papyrus large scale installation projects. In 2012 he started the ISIS Papyrus Academy to stimulate scientific research projects like CACAO in cooperation with the University of Vienna.

Christoph Czepa is a researcher at the Faculty of Computer Science, University of Vienna, Austria. His research areas include Business Process Management (BPM), Adaptive Case Management (ACM), Domain-Specific Languages (DSLs), software architecture, software engineering, compliance/consistency and the application of machine learning and formal verification methods in the aforementioned domains. He has received a master degree in computer science in December 2013 (with distinction). Currently, he is pursuing a PhD in computer science. Christoph is the (co-)author of more than 15 peer-reviewed scientific articles and papers, and he participated in two research projects, namely the CACMTV (Content-Aware Coding for Mobile TV) project and CACAO (Consistency Checking, Recommendations, And Visual Modeling For Ad Hoc Changes By Knowledge Workers In Adaptive Case Management) project.

Huy Tran is a senior postdoc working at the Research Group Software Architecture (SWA), Faculty of Computer Science, University of Vienna, Austria. Before that Huy received his doctoral degree in December 2009 and worked as a postdoc at Vienna University of Technology, Austria. He has published some 40 articles and papers in peer-reviewed journals, conferences, book chapters, and workshops. He has also taken part in several European and domestic R&D projects since 2008. His current research interests are in the field of software engineering including software architecture, view-based architecture, model-driven engineering, service-oriented computing, distributed event-based software systems, business process management, business process compliance, and applied formal methods in software engineering.

Uwe Zdun is a full professor for software architecture at the Faculty of Computer Science, University of Vienna. Before that, he worked as assistant professor at the Vienna University of Technology and the Vienna University of Economics respectively. He received his doctoral degree from the University of Essen in 2002. His research focuses on software design and architecture, empirical software engineering, distributed systems engineering (service-based, cloud, mobile, and process-driven systems), software patterns, domain-specific languages, and model-driven development. Uwe has published more than 210 articles in peer-reviewed journals, conferences, book chapters, and workshops, and is co-author of the books "Remoting Patterns", "Process-Driven", and "Software-Architektur." He has participated in 26 R&D projects. Uwe is editor of the journal *Transactions* on Pattern Languages of Programming (TPLoP) published by Springer, Associate Editor of the Computing journal published by Springer, and Associate Editor-in-Chief for design and architecture for the IEEE Software magazine.