A Framework for Safety-critical Process Management in Engineering Projects*

Saimir Bala[1], Cristina Cabanillas[1], Alois Haselböck[2], Giray Havur[1]
Jan Mendling[1], Axel Polleres[1], Simon Sperl[2], and Simon Steyskal[1,2]

[1]Vienna University of Economics and Business, Austria
{name.surname}@wu.ac.at
[2]Siemens AG Österreich, Corporate Technology, Vienna, Austria
{name.surname}@siemens.com

Abstract. Complex technical systems, industrial systems or infrastructure systems are rich of customizable features and raise high demands on quality and safety-critical aspects. To create complete, valid and reliable planning and customization process data for a product deployment, an overarching engineering process is crucial for the successful completion of a project. In this paper, we introduce a framework for process management in complex engineering projects which are subject to a large amount of constraints and make use of heterogeneous data sources. In addition, we propose solutions for the framework components and describe a proof-of-concept implementation of the framework as an extension of a well-known BPMS.

Keywords: Adaptation, compliance, engineering, process management, resource management, unstructured data

1 Introduction

Deployments of technical infrastructure products are a crucial part in the value-creation chain of production systems for large-scale infrastructure providers. Examples of a large-scale, complex engineering process in a distributed and heterogeneous environment are the construction of a railway system comprising electronic interlocking systems, European train control systems, operator terminals, railroad crossing systems, etc.; all of the systems are available in different versions using a variety of technologies. It is often necessary to offer, customize, integrate, and deliver a subset of these components for a particular customer project, e.g. the equipment of several train stations with an electronic switching unit in combination with a train control system based on radio communication for a local railway company. Configurators are engineering tools for planning and customizing a product. Each subsystem comes with its own, specialized engineering and verification tools. Therefore, configuring and combining these

* This work is funded by the Austrian Research Promotion Agency (FFG) under grant 845638 (SHAPE): http://ai.wu.ac.at/shape-project/

subsystem data to a coherent and consistent system has to follow a complex, collaborative process.

A challenge is the management and monitoring of such complex, yet mostly informally described, engineering processes that involve loosely integrated components, configurators and software systems [1]. Nowadays, many of the steps required (e.g. resource scheduling, document generation, compliance checking) tend to be done manually, which is error-prone and leads to high process execution times, hence potentially affecting costs in a negative way.

In this paper, we explore this domain and present a framework for process management in complex engineering processes that includes the formalization of human-centric process models, the integration of heterogeneous data sources, rule enforcement and compliance checking automation, and adaptability, among others. The framework has been defined from an industry scenario from the railway automation domain. Furthermore, we describe solutions to support the functionalities required by every framework component as well as a proof-of-concept implementation of the framework that can be integrated with an existing Business Process Management System (BPMS). The goal is to help to develop ICT support for more rigorous and verifiable process management.

The rest of the paper is organized as follows. Section 2 delves into the problem and derives system requirements. Section 3 presents the framework and solutions for its components. Section 4 describes a proof-of-concept implementation. Section 5 summarizes related work. Finally, Section 6 draws conclusions from this work and outlines ideas for future extensions.

2 Motivation

In the following, we describe an industry scenario that exemplifies the characteristics of complex engineering processes and define a set of system requirements for the challenges identified in it.

2.1 Industry Scenario

Activities to create complete, valid and reliable planning, and customization process data for a product deployment are part of an overarching engineering process that is of crucial importance for the success of a project in a distributed, heterogeneous environment. Fig. 1 depicts a generic engineering process for building a new infrastructure system in the railway automation domain modeled with Business Process Model and Notation (BPMN) [2].

The engineering process itself is represented in the pool *Railway automation unit* and comprises the building and testing of the system. The pool *Resource planning unit* as well as the activities depicted in gray represent a meta-process comprising scheduling activities that are performed in the background in order to enable the completion of the engineering process in compliance with a set of restrictions (temporal and logistics, among others) while making an appropriate use of the resources available. Resource allocation is of great importance to

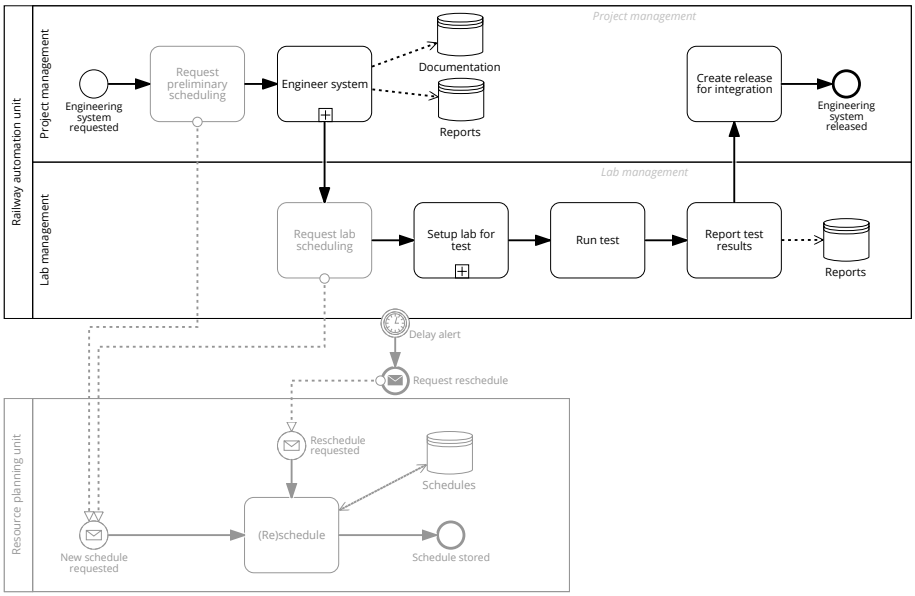

Fig. 1: Generic engineering process in the railway automation domain

large-scale engineering processes in which a large variety of different resources, ranging from laboratories and specific hardware to engineers responsible for the correct execution of the process, are involved and unexpected situations may have critical consequences (e.g., delays resulting in unplanned higher costs).

Hence, the first step consists of scheduling the building of the system. Building the system is, in turn, a process composed of several activities (potentially operating on different levels of abstraction) each involving a large variety of different resources, data sources, and data formats used. Specifically, the customer provides input data in form of, e.g., XML documents representing railway topology plans, signal and route tables, etc., which are used by the engineers to configure the product. Typically, several configuration tools are involved in that process too, complemented by version control and documentation activities. The result is a set of data of various kinds and formats (i.e., XML, JSON, and alike) such as bill of material (BOM), assembly plans, software configuration parameters, and all other documents and information required for the testing, integration, and installation of the system. Additionally, we map all gathered data to a common extendable RDF model in order to make use of standard data integration and processing strategies from the Semantic Web (e.g., OWL, SPARQL, SHACL, etc.). The engineering project manager orchestrates and monitors these engineering tasks. Besides, further data is generated during the execution of the subprocess *Engineer system* in the form of, e.g., emails exchanged between the process participants.

Once the system is built, it must be tested before it is released for its use. That procedure takes place in laboratories and comprises two phases: the test setup

and run phases. Like before, it is necessary to schedule these activities taking into consideration the setting and all the restrictions for the execution of the activities. The setting is the following: there are several space units distributed into two laboratories and several units of different types of hardware for conducting the testing. The employees of the organization involved in these activities are specialized in the execution of specific testing phases for specific types of systems, i.e. there may be an engineer who can perform the setup for a system S_1 and the test execution for a system S_2, another engineer who cannot be involved in the testing of system S_1 but can perform the two activities for the system S_2, and so on. As for the restrictions, they are as follows: each task involved in these two phases requires a specific set of resources for its completion. In particular, the setup tasks usually require one employee working on one unit of a specific type of hardware in a laboratory, and the run activity usually requires several employees for its execution. Besides, a test can only be executed if the whole setup takes place in the same laboratory. In addition, for the scheduling it is necessary to take into account that other instances of the same or different processes might be under execution at the same time and they might share resources.

The setup and the run test activities will then be executed according to the plan. Similar to the engineering step, data comprising the results of the tests, emails, Version Control System (VCS) file updates and the like, is generated during the testing steps. Railway projects also generate other types of data which play a role in the running system, e.g., cut plans, signals form the tracks, actual user data of the system, etc. The latter can be useful when it comes to monitoring safety-critical processes during their execution. Given the great number of software tools involved in the process, our scenario focuses more on the software engineering aspect of the railway domain. Hence, we use VCS logs to track the evolution of the artifacts that are produced during such software engineering process. Nevertheless, it can be extended towards all kinds of artifacts that are stored in VCSs, such as the different versions of outputs from engineering tools.

When the testing of the system is finished, a final report is written and archived with the information generated containing the description of the test cases, test data, test results, and the outline of the findings. Responsible for the final version of this report is the testing project manager. Finally, the engineering project manager deploys a complete and tested version of the engineering system and the integration team takes over the installation of the product.

Note that unexpected situations may cause delays in the completion of any of the activities involved in the engineering process. It is important to detect such delays as soon as possible in order to properly schedule the use of resources and figure out when the process can be finished under the new circumstances. Therefore, rescheduling may be required at any point, involving all the aforementioned restrictions and possibly new ones.

2.2 Challenges and Technical Requirements

A number of issues are involved in the industry scenario previously described when it comes to automating its execution. From the analysis of the process description, the following challenges have been identified:

Challenge 1: Integrated description of processes, constraints, resources and data. Operating with processes like the one described before implies taking not only the order of execution of the process activities and behavioural constraints typically enforced in the process model into consideration, but also information related to other business processes perspectives, such as resources and data requirements, as well as regulations affecting, e.g., the use of these. Several formal languages are at hand for describing processes [2], constraints [3], resources [4] and data (e.g. XML) separately. However, a challenge is to define all them in an integrated manner with a model that provides rich querying capabilities to support analysis automation, status monitoring or respectively, the verification of constraints and consistency.

Therefore, a system for automating processes like the engineering process would require *an integrated semantic model to describe and monitor processes, resources, constraints and data (RQ1).*

Challenge 2: Integration and monitoring of structured and unstructured data. To a high degree, engineering steps are the input for state changes of the process, often only visible as manipulation of data. Hence, a engineering process must also incorporate these data smoothly for monitoring control flow, version updates, data storage, and email notification. To this end, various types of systems have to be integrated including their structured (e.g., logs from tools, or databases) and unstructured data (e.g., by mail traffic, or ticketing systems). Up until now, these data are hardly integrated and are monitored mostly manually.

Therefore, techniques to gather relevant information from unstructured or semi-structured data sources, such as emails, VCS repositories and data from tools, and transform them into understandable data structures must be put in place, i.e. a system for automating processes like the engineering process would require *mechanisms to detect and extract structured from unstructured process data (RQ2).*

Challenge 3: Documentation of safety-critical, human and data aspects and compliance checking. Engineering projects have time-critical phases typically prone to sloppy documentation and reduced quality of results. Many of the process steps are required to be documented in prescribed ways by standards and regulations (e.g. SIL [5]). Considerable amount of time is spent in the manual documentation of process steps as well as in the integration of the documentation of separate modules for generating final reports. Furthermore, in such safety-critical environments the use of resources must be optimized and rules must be enforced and their fulfillment ensured. For instance, in our industry scenario we can observe a typical series of data management steps, including: check in a new version of a data file, inform the subsequent data engineer, confirm and document this step in the process engine, etc. The latter two steps could be done automatically once the process engine has detected the check-in into the VCS.

This automation would also lead to a significant decrease of the overall execution time as well as a potential reduction in the number errors typically caused by human mistakes.

Therefore, a system for automating processes like the engineering process would require *a method for flexible document generation (RQ3)* as well as *reasoning mechanisms (RQ4)* and *monitoring capabilities (RQ5)* for automating resource allocation and compliance checking.

Challenge 4: Be ready for changes. Despite engineering process definitions are quite stable and might remain unchanged for a long time, an automatic monitoring and a thorough analysis of process models and executions may lead to the discovery of potential improvement points to make processes simpler and less error-prone. Similarly, changes in the schedule of activities and resources can be necessary at any time due to a number of reasons including delays or unexpected unavailability of resources, among others. Currently, all these adaptations require manual work and are prone to errors.

Consequently, methods to detect and deal with changes must be put in place. This might include the monitoring of process executions and the analysis of the generated execution data (e.g. with process mining techniques) to anticipate delays as well as the need of having available mechanisms capable of reallocating the resources according to changeable requirements and circumstances. Therefore, besides requirements RQ4 and RQ5, a system for automating processes like the engineering process would also require *adaptation procedures to react to changing conditions (RQ6)*.

Challenge 5: Acceptance and human factors. The overall process management needs to be set up in a non-obtrusive way, such that engineers executing the processes find it useful and easy to use. This is a specific challenge in safety-critical systems, which are developed with a tight timeline. It calls for a design that integrates existing tools and working styles instead of introducing new systems.

Therefore, the automation of processes like the engineering process would require *an integrated system (RQ7)* that provides all the functionality, which involves general features of a BPMS (e.g. process modeling and execution) extended to support the demands of safety-critical, human- and data-centric processes as described in our industry scenario.

3 Framework

We have designed a framework that provides the support required to address the challenges identified in complex engineering processes. It consists of a data model and five functional modules that interact with a BPMS, as depicted in Fig. 2 using the Fundamental Modeling Concepts (FMC) notation[1].

In order to support *RQ1*, a semantic model encompassing the various types of domain data that must be represented and manipulated must be defined. Hence, this model stores all static and dynamic data used by the BPMS and by the

[1] `http://www.fmc-modeling.org/`

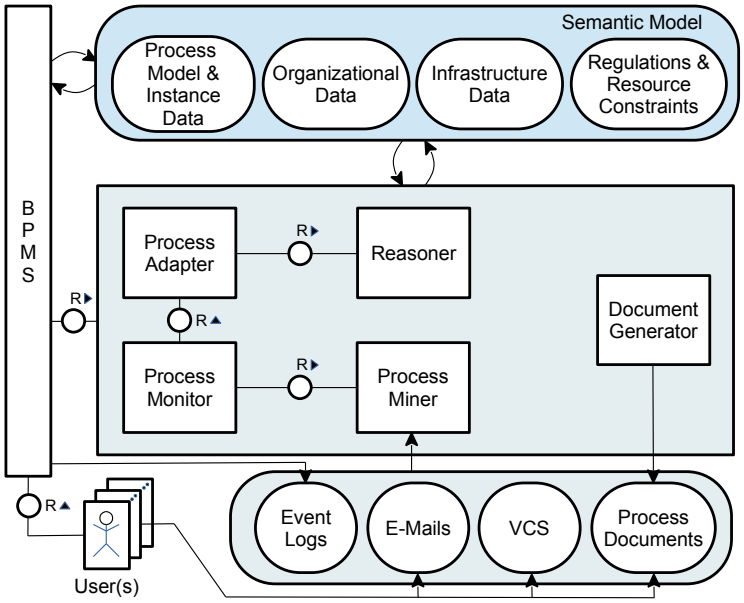

Fig. 2: Proposed framework for process management in complex engineering projects

functional components. Specifically, these data include: process models and their instances, organizational data related to human resources, infrastructure data related to non-human resources, and constraints derived from regulations and norms (e.g. SIL) as well as further requirements related to the utilization of resources. The semantic model implicitly operates as a communication channel between the BPMS and all the functional modules and hence, all of them must have read&write access to the model.

Typical functionality of a BPMS include modeling and executing processes. Information about process instances is usually stored in event logs generally including, among others, temporal and resource information related to the execution of the process activities [6]. In addition to that structured information, as described in Section 2.1, several kinds of unstructured and semistructured data are generated during the execution of complex engineering processes, e.g. emails, VCS files and reports. All the data produced during process execution must be analyzed in order to detect anomalies (e.g., deviations from the expected behavior).

The *Process Miner* component of our framework tries to discover as much data relevant to the current state of a process execution as possible, performs the transformations required as specified by *RQ2*, and communicates the information extracted to the *Process Monitor* (*RQ5*) periodically under request. In case the *Process Monitor* reveals a discrepancy between process instance data and the data discovered by the process miner (e.g., a delay), it informs the *Process Adapter*

about the discrepancy. The *Process Adapter* analyzes the deviation and responds by proposing an adaptation solution to the BPMS in order to put the process back into a coherent and consistent state, as specified in *RQ6*. The adaptation may consist of small changes that can be performed directly on the BPMS side or, on the contrary, of complex recovery actions that may require reasoning functionalities. In the latter case, the *Reasoner* comes into play by, e.g., doing a new activity or resource scheduling according to the new domain conditions. Therefore, the *Reasoner* can be seen as a supportive component that helps the BPMS with typical activities, such as the scheduling of process activities, and the allocation of resources to those activities in accordance with resource constraints and regulations defined in the semantic model. This covers *RQ4*.

Finally, the *Document Generator* of the framework provides support for *RQ3* by helping to fill out the documents that must be generated as output of process activities. As mentioned before, this automation is expected to decrease reporting errors, especially in documents related to auditing.

The design of the framework as an extension of the functionality present in existing BPMSs attends to *RQ7* and hence, it intends to increase the acceptance by users familiar with Business Process Management (BPM).

In the following, we describe our solution for the implementation of the functionality provided by the most domain specific components of the framework.

3.1 Semantic Model

Aiming at automation, we believe that Semantic Web technologies provide the most appropriate means for (i) integrating and representing domain-specific (heterogeneous) knowledge in a consistent and coherent format, and (ii) querying and processing integrated knowledge.

Therefore, following the METHONTOLOGY approach [7], we have developed an engineering domain ontology [8] that integrated three different domains of interest relevant for our approach, namely: (i) engineering domain and organizational (i.e. resource-related) knowledge; (ii) business processes; and (iii) regulations and policies [9].

Representing Infrastructural & Organizational Knowledge One of the first steps for developing an ontology according to the METHONTOLOGY approach involves the definition of an *Ontology Requirements Specification Document* [10]. In order to address the requirements gathered throughout that process we decided to adopt parts of the organizational meta model described in [11] and enriched it with concepts for modeling teams [12] (cf. Fig. 3) for representing infrastructural & organizational knowledge. Using these two meta models presents an advantage. Specifically, the organizational meta model described in [11] has been used to design a language for defining resource assignment conditions in process models called Resource Assignment Language (RAL) [4]. As was shown in [13], that language can be seamlessly integrated in existing process modeling notations, such as BPMN, thus enriching the process models with expressive

resource assignments that cover a variety of needs described by the creation patterns of the well-known workflow resource patterns [11]. Furthermore, a graphical notation was later designed with the same expressive power as RAL in order to help the modeler to define resource assignments in process models [14]. The meta model for teamwork assignment was also considered to develop an extension of RAL called RALTeam [12], which, however, lacks a graphical notation so far. Therefore, if support for these expressive notations were introduced in the BPMS, the ontology would support them at the same time as it supports less expressive means of assigning resources to process activities (e.g. based on organizational positions).

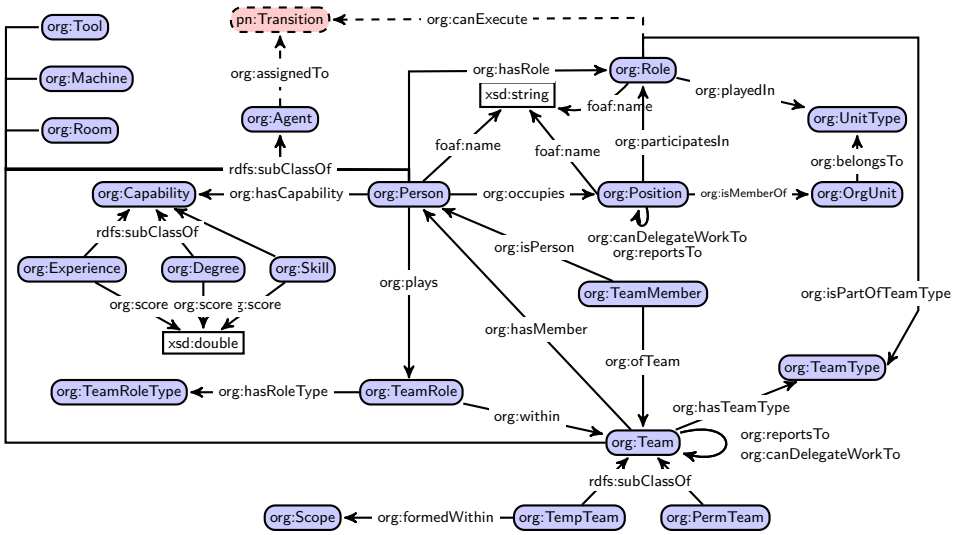

Fig. 3: Ontology for infrastructural & organizational knowledge.

Representing Business Processes Driven by the requirements of our resource allocation approach (cf. Section 3.2), we decided to transform BPMN models into timed Petri nets [15] as an intermediary format for reasoning tasks (e.g., scheduling [16]) by using the transformation proposed in [17, 18], and store these Petri nets in our ontology. Note that the user only interacts with the BPMN model while using the system as we use the timed Petri net representation internally. There are several reasons for using Petri nets for process modeling [19], namely:

– *Clear and precise definition:* Semantics of the classical Petri net is defined formally.

– *Expressiveness:* The primitives needed to model a business process (e.g. routing constructs, choices, etc.) are supported.
– *Tool-independent:* Petri nets have mappings to/from different business modelling standards [20]. Moreover, this immunizes our ontology from changes in business modeling standards.

For modeling Petri nets themselves we adopted selected concepts of the Petri Net Markup Language (PNML) [21] and represented them in terms of an RDFS ontology (cf. Fig. 4).

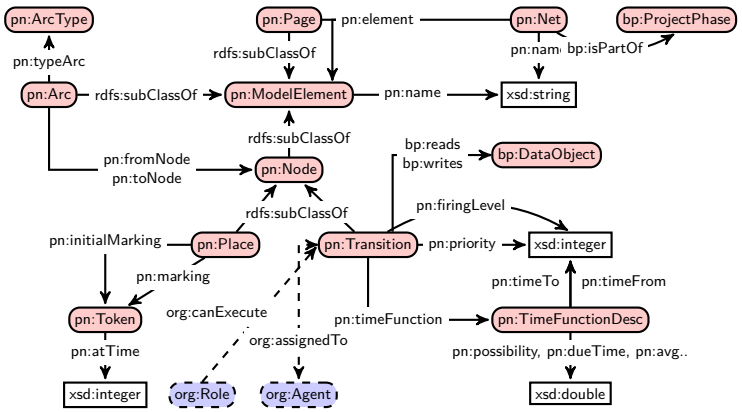

Fig. 4: Ontology for representing processes and process instances.

Extracting and Specifying Compliance Rules One of the most important aspects of dealing with safety-critical human- and data-centric processes is providing means for proving that business processes comply with relevant guidelines such as domain-specific norms and regulations, or workflow patterns. As illustrated in Fig. 5 and described in [22], establishing proper compliance checking functionalities typically requires to extract and interpret a set of *Compliance Objectives* from respective *Compliance Requirements* first, before those objectives are specified in terms of *Compliance Rules/Constraints* using an appropriate specification language (i.e. a language capable of representing all types of compliance rules/constraints relevant for the respective domain of interest). Specified compliance rules and constraints are then subsequently used by a monitoring/-compliance checking engine for verifying correct and valid execution of business processes w.r.t. previously defined rules.

1. *Identifying Compliance Objectives:* Organizations have to deal with an increasing number of norms and regulations that stem from various compliance sources, such normative laws and requirements. In the railway domain processes have to be compliant with specific European Norms (i.e. 50126, 50128, and 50129). For example, EN50126 defines guidelines for managing Reliability, Availability, Maintainability, and Safety (RAMS) of safety-critical business

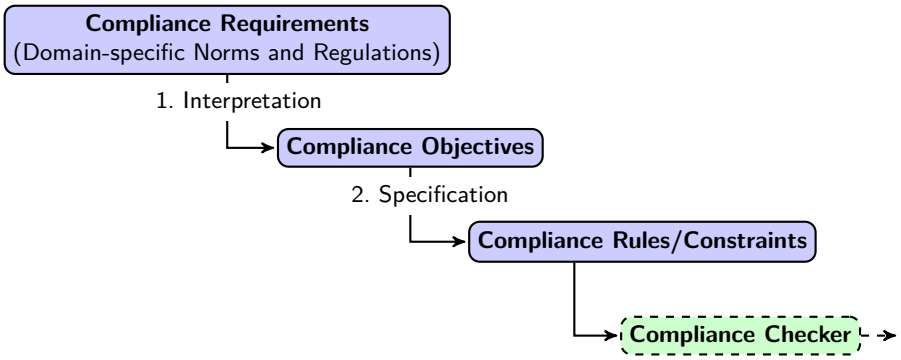

Fig. 5: General approach for business process compliance monitoring [22].

processes, where we extracted objectives, requirements, deliverables, and validation activities defined in all phases of the RAMS lifecycle [23, 24].

2. *Representing Compliance Rules and Constraints:* Since all process relevant data are stored in RDF, we plan to utilize recent advancements in the area of constraint checking for RDF, i.e. the Shapes Constraint Language (SHACL) [25] for representing and validating identified compliance objectives. Since constraints expressed in SHACL are internally mapped to corresponding SPARQL queries, we can further complement our own compliance constraints with already existing approaches for compliance checking using SPARQL such as [26]. Compliance constraints expressed in SHACL are tightly integrated with the underlying ontology and can be validated during both design time and runtime[2].

3.2 Reasoner

The *reasoner* module supports our framework on top of the engineering domain ontology in two folds: (i) by performing automated resource allocation described in the declarative formalism Answer Set Programming (ASP) [27], and (ii) by querying the ontology for compliance checking. We also looked at other declarative programming paradigms (e.g., CLP(FD)), and our initial findings confirm the advantages of using ASP [28, 29]. Some of these advantages are as follows:

- Compact, declarative and intuitive problem encoding
- Rapid prototyping and easy maintenance (e.g., no need to define heuristics)
- Complex reasoning modes (e.g., weight optimization)
- Ability to model effectively incomplete specifications
- Efficient solvers (e.g., *clingo*)

[2] For a more detailed introduction on utilizing SHACL for defining custom constraints, we refer the interested reader to [25].

In the literature, ASP is preferable when the size of the problem does not explode the grounding of the program [29, 30]. We show that our resource allocation encoding in ASP is applicable to the problems of business processes at a real-world scale [16].

Resource Allocation

Resource allocation aims at scheduling activities of a business process and properly distributing available resources among scheduled activities. We address the problem of allocating the resources available to the activities in the running process instances in a time optimal way, i.e. process instances are completed in the minimum amount of time. Therefore, our resource allocation technique makes business process executions effective and efficient.

We encode the resource allocation problem in Answer Set Programming (ASP) [27], a declarative (logic programming style) paradigm. Its expressive representation language, efficient solvers, and ease of use facilitate implementation of combinatorial search and optimization problems (primarily *NP-hard*) such as resource allocation. Therefore, modifying, refining, and extending our resource allocation encoding is uncomplicated due to the strong declarative aspect of ASP. We use the ASP solver *clasp* [27] for our purpose as it has proved to be one of the most efficient implementations available [31]. Another complex reasoning extension supported in *clasp* are weight optimization statements [27] to indicate preferences between possible answer sets.

Resources are defined in the engineering domain ontology (the organizational data and the infrastructure data) where they are characterized by a *type* and can have one or more *attributes*. In particular, any resource type (e.g., org:Person in Fig. 3) is a subclass of org:Agent. The attributes are all of type rdf:Property. The organizational data consists of human resources, their attributes (e.g. their name, role(s), experience level, etc.) and current availabilities stored in the ontology. In the same fashion, infrastructure data represents material resources (i.e. tools, machines, rooms) and their availabilities. Resource allocation considers resources to be *discrete* and *cumulative*. Discrete resources are either fully available or fully busy/occupied. This applies to many types of resources, e.g. people, software or hardware. However, for certain types of infrastructure, availability can be partial at a specific point in time. For instance, a room's occupancy changes over time. Such a cumulative resource is hence characterized by its *dynamic* attribute (available space in the room) and it can be allocated to more than one activity at a time. Any statement in our ontology can be easily incorporated as the input of our problem encoding [32]. The following example shows an excerpt of organizational data in the ontology and its equivalent in ASP.

```
# Organizational data
:glen a org:Person; foaf:name "Glen";
      org:occupies testeng.
:testeng a org:Position; foaf:name "Test Engineer";
         org:participatesIn labmng.
```

```
:labmng a org:Role; foaf:name "Lab Management".

% Equivalent ASP encoding
person(glen). name(glen,"Glen").
occupies(glen,testeng).
position(testeng). name(testeng,"Test Engineer").
participatesIn(testeng,labmng).
role(labmng). name(labmng,"Lab Management").
```

There are two main operations under resource allocation: *Allocation of resources* and *re-allocation of resources as adaptation*.

Allocation of resources deals with the assignment of resources and time intervals to the execution of process activities. It can be seen as a two-step definition of restrictions. First, the so-called *resource assignments* must be defined, i.e., the restrictions that determine which resources can be involved in the activities [4] according to their properties. The outcome of resource assignment is one or more *resource sets* with the set of resources that can be potentially allocated to an activity at run time. The second step assigns cardinality to the resource sets such that different settings can be described.

As mentioned in Section 3.1, there exist languages for assigning resource sets to process activities [4, 33–35]. However, cardinality is generally disregarded under the assumption that only one resource will be allocated to each process activity. This is a limitation of current BPMS, which we overcome in our proposed framework.

The main temporal aspect is determined by the expected duration of the activities. The duration can be predefined according to the type of activity or calculated from previous executions, usually taking the average duration as reference. This information can be included in the executable process model as a property of an activity (e.g. with BPMN [2]) or can be modelled externally. As for the variable activity durations depending of the resource allocation, three specificity levels can be distinguished:

- *Role-based duration*, i.e., a triple $(activity, role, duration)$ stating the (minimum/average) amount of time that it takes to the resources within a specific resource set (i.e., cardinality is disregarded) to execute instances of a certain activity.
- *Resource-based duration*, i.e., a triple $(activity, resource, duration)$ stating the (minimum/average) amount of time that it takes to a concrete resource to execute instances of a certain activity.
- *Aggregation-based duration*, i.e., a triple $(activity, group, duration)$ stating the (minimum/average) amount of time that it takes to a specific group to execute instances of a certain activity. In this paper, we use *group* to refer to a set of human resources that work together in the completion of a work item, i.e., cardinality is considered. Therefore, a *group* might be composed of resources from different roles which may not necessarily share a specific role-based duration. An aggregation function must be implemented in order

to derive the most appropriate duration for an activity when a group is allocated to it. The definition of that function is up to the organization.

Given (i) a process model and its instance data; (ii) organizational and infrastructure data; (iii) resource requirements, i.e. the characteristics of the resources that are involved in each activity to be allocated (e.g. roles or skills); (iv) temporal requirements; and (v) regulations such as access-control constraints [4], i.e. *separation of duties (SoD)* and *binding of duties (BoD)*, the ASP solver finds an optimal allocation. The aforementioned functionalities and the entire associated ASP encoding are detailed in [36].

While executing the process instance, changes may be introduced to input used for allocation. For instance, organizational data may change in case of absence, regulations may be modified or simply execution of activities may delay. In some cases, such a change in the ontology directly affects a running process instance, and therefore, the process monitor informs the process adapter. The process adapter may decide that the allocation should be performed again. *Adaptive re-allocation* is a key functionality in this scenario and it is indispensable for safety-critical, human- and data- centric process management.

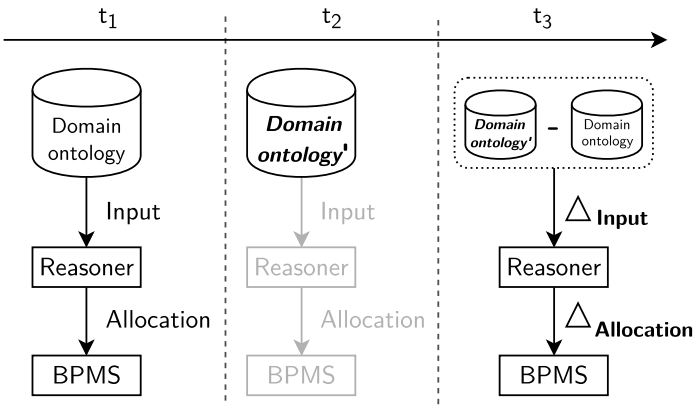

Fig. 6: Adaptive re-allocation timeline

Fig. 6 shows this scenario in three consecutive time steps: After allocating resources to a process instance at t_1, some changes are introduced at t_2 that interfere with the original allocation, and hence, an adaptive reallocation is performed at t_3. The reasoner computes a delta allocation, i.e. the original allocation is preserved as much as possible. Therefore, some activities might be rescheduled, and others might be shifted and/or reallocated to some different resources in a *minimal* fashion.

Compliance Checking As mentioned previously, we define compliance constraints over business processes using SHACL. In order to do so, we translate each

compliance objective to a corresponding SPARQL query first, before embedding it in a respective constraint component, which itself can then be integrated into the ontology [24].

3.3 Process Monitor and Process Adapter

Changes and deviations to the processes may occur during execution. For instance, new rules and regulations may require the process to operate differently. We want our framework to be able to handle these unexpected events.

The process data and the evidence from the process miner are compared by the process monitor for detecting deviations. The main idea is that the process adapter is informed about the deviations, therefore it minimizes the impact of these deviations in the running process instances by offering a recovery strategy. The process monitor and process adapter address the requirements *RQ5* and *RQ6*, respectively. The process monitor is able to run several algorithms for monitoring both the process behaviour and the process compliance to rules and regulations. This is performed by checking the current process constraints against the data from our semantic model.

The process adapter is in charge of handling exceptions that arise from the process monitor. This component acts in two different ways: Either it *i)* corrects process behaviour with minimal intervention, or, in case a more complex adaptation is required, *ii)* it stops the process and notifies the reasoner for planning an adaptation.

3.4 Process Miner

Traditional process mining algorithms [6, 37] are able to give valuable insights into the different perspectives of a business process. Process models inferred from log files can further be analyzed for bottlenecks, performance, deviations from the expected behaviour, compliance with rules and regulations, etc. Regardless of the perspective they aim at mining as well as the type of process modeling notation used to represent the outcome (declarative versus imperative process mining), all these process mining algorithms require properly structured data. Specifically, they must comply with the XES [38] meta model. Any of the existing process mining techniques is a candidate to be used for the implementation of the *Process Miner* component in regard to the functionality related to traditional process mining and the decision should be made according to the specific characteristics desired.

However, the biggest challenge of this component in our framework is to deal with unstructured and semistructured data generated in the execution of the process activities, generally in the form of VCS files and emails. Although it is hard to mine process models out of such unstructured or semistructured data, some approaches can be used to obtain valuable insights on them. Specifically, [39] allow for transforming semi-structured VCS logs to process activities which can be mined by classic mining algorithms. Poncin et al. [40] developed the FRASR framework, which is enables to answers engineering questions by applying process

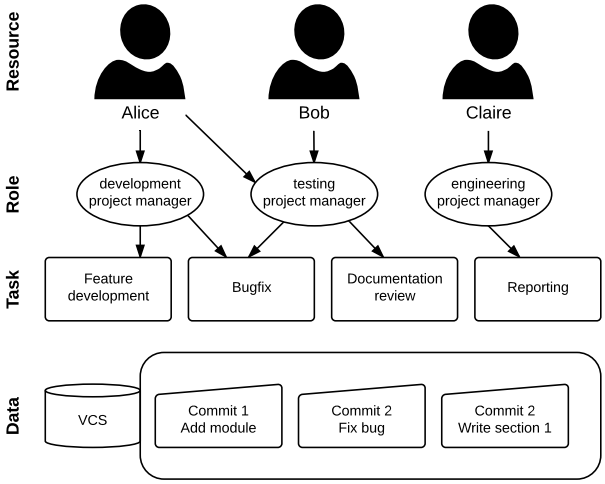

Fig. 7: Software project and resources

to software repositories. The challenge here is to identify the relevant events for the files, from a process mining point of view. Di Ciccio et al. [41] propose the MailOfMine approach to discover artful processes laying underneath email collections. Bala et al. [42] adopt a visualization approach by mining project Gantt charts from VCS logs.

Driven by the fact that complex engineering process like the industry scenario described in Section 2.1 are resource-intensive, we have developed a novel approach to extract organizational roles from VCS repositories. VCSs have both structured and unstructured data. On the one hand, they explicitly provide information about the user who performed changes in some file(s) and the time at which she committed the new version(s). On the other hand, they have a textual part typically carrying a comment that explains the changes performed on the file(s). Note that these kind of data are similar to data from email. In fact, both emails and git comments have in common information about the user, the timestamp, and a textual description. Moreover, we use an ontology (cf. Section 3.1) to store mining insights. That is, we allow for the integration of all types of data that can be represented in RDF, including data coming from engineering tools. There are indeed tools in SVN [43] or Git [44] that allow for sending user commits as emails. Therefore, discovering roles out of such data (and especially when the outcome is combined with the result of mining activities) might help the *Process Monitor* to identify potential deviations regarding the resources that have actually performed specific tasks or manipulated certain information. Hence, it contributes to compliance checking.

Let us see an example of users who use VCS to collaborate on a software development project. Fig. 7 shows a setting with three users named Alice, Bob and Claire. Alice is a development project manager. She works with her colleagues

Bob and Claire. Alice is mainly responsible for *feature development*, but she is also involved in *testing* project-management team. Her tasks include the *development* of new features and fixing of related bugs. In her first *commit* she adds a new message where she describes her work. The message to describe her changes is "added new module to demo". In the first row of Table 1, identified by *commit id 1*, Alice's change is reported. Bob is part of the *testing* project-management team. His task is to ensure that the code submitted by the development team complies with existing standards and contains no errors. He discovers and fixes some minor bugs in Alice's code and informs Alice on further work needed on the analyzed features. Meanwhile, he commits his changes with *commit id 2* and comments "Modified the setup interface". Consequently, Alice reworks her code and commits a new version as reported in row 3 of Table 1, commenting her work with "Update application interface". As an *engineering project manager*, Claire takes over and starts to work on the documentation. She commits part of her work as in row 4. As the project continues, the work is accordingly stored in the log as shown in the table.

Id	User	Timestamp	Changed	Comment
1	Alice	2014-10-12 13:29:09	Demo.java rule.txt	Added new module to demo and updated rules
2	Bob	2014-11-01 18:16:52	Setup.exe	Modified the setup interface
3	Alice	2015-06-14 09:13:14	Demo.java	Update the application interface
4	Claire	2015-07-12 15:05:43	graph.svg todo.doc	Define initial process diagram & listed remaining tasks
...	...	...	...	...

Table 1: Example of a VCS log

Our approach leverages both on the file types and the comments of the users. We devise an algorithm that classifies users into a set of roles. For that purpose, we approach the role discovery problem as a classification problem. We define two methods: one based on user clustering and one based on commit clustering. A prior step for this is the feature selection. By looking at the commit data, we identify the following features:

- Total number of commits.
- Timeframe between the first and the last commit of a user (i.e. the time he has been working on the project).
- Commit frequency: total number of commits divided by the time frame.
- Commit message length: average number words in the commit comment.
- Keyword count: how often determinate words like "test" or "fix" are used
- Number of files changed.
- Affected file types: how often a file with a certain format (e.g., *.java, *.html) are modified by a user, relative to the total number of modified files

Then, we use the features for two machine learning algorithms. In the first approach we iterate through the users and cluster them using the k-means algorithm. Consequently we build classification models using decision trees. We then train three different datasets individually and cross validate the results. The second approach starts from the commits. The main idea here is that we do not want to assign users to a specific category. Rather, we allow for users having multiple roles and classify their contribution in each commit. We build user profiles that account for fractions of contributions of each user to the different classes. Classes used in the classification for the example described above would be: *Test, Development, Web, Backend, Maintenance, Refactor, Documentation, Design, Build, Data, Tool, Addition, Removal, vcsManagement, Automated, Merge*. Each commit is classified into one of the classes according to its features. Users who committed can be then classified by their commits. The classification can be done both manually and automatically: *i)* rules can be manually inferred by looking for similarities between users with the same role; or *ii)* an automated classification can be performed by using machine learning algorithms. For example, decision trees can be used for an automated classification. In this case the *commit* type percentages are used as features and the manually assigned roles as classes. As a further step, the resulting decision tree models from the different datasets can be cross-validated. The complete approach and its evaluation can be found in [45].

3.5 Document Generator

Safety-critical engineering systems require well-documented process steps. Engineers are in charge of clearly describing their tasks in such a way that it is possible to audit their work. These documents are often manually created. This has at least four drawbacks. First, their creation is laborious. Second, it is error-prone and misaligned in terms of language. Third, it is described at different levels of granularity. Fourth, it is difficult to process and audit afterwards.

A simple example is the following. Engineers need to work on a specific task and use predefined tools. Their tasks and their version tools are specified at the beginning of the project and must be consistent during the project's lifetime. Tool versions must usually be filled in the documentation generated, e.g. in reports. In a big engineering project, tools can be numerous and their versions are far from being user-friendly. This makes the risk of human mistakes very likely.

To assist engineering project managers in producing audible documentations, we have developed a customizable approach for partially automating document generation. In particular, it is able to fill in trivial information (e.g., tool version, user name, task to which the user is assigned, etc) into word processor documents. Our document generation technique is based on templates. These templates consist of evolving documents and are automatically filled in during the workflow, and therefore enable flexible process verification. Our approach generates standard documents which are compatible with predefined word processor programs and can be opened and edited by them.

Document generation comprises four steps, depicted in Figure 8 and explained next. A mapping function is first defined from a process activity to a document

which is generated as its outcome. Afterwards, an interpreter is defined which is in charge of filtering the relevant process activities and variables. Process variables are used by the BPMS during the execution of the process. Examples of process variables can be the name of the user that is currently assigned to one activity, the name of the running process instance, and everything that adds data to the executing process in the BPMS. The interpreter is not strictly bound to a particular process nor to a particular template, and is defined externally. This supports changes both in the process and in the template. The writing in the document is triggered by a listener. A listener waits for activity events. As soon as an activity is submitted, the interpreter and the mapping function work together to generate *(variable key,value)* pairs in the document template. This is run iteratively on the document until all the trivial data is filled in.

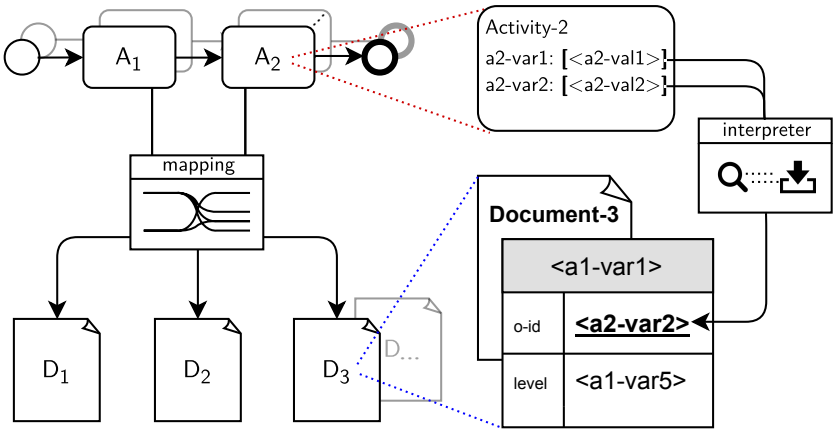

Fig. 8: Implementation showing how form values (process variables) from Camunda could look like in a generated document

4 Proof-of-concept Implementation

As a proof of concept we have implemented the main components of the framework discussed in Fig. 2. In this prototype, we aim to bring together functionality from reasoning, process mining and document generation. Fig. 9 shows the software architecture that we use. It considers four main components which interoperate during the execution of a process activity.

Here, we describe the main components of the architecture and their interactions.

Camunda running process. We use the Camunda BPM engine as our BPMS. Camunda is an open source platform that allows for defining new components and for interacting with its APIs in a custom way. All the process instances that run into Camunda and their data are stored in log files. Camunda uses

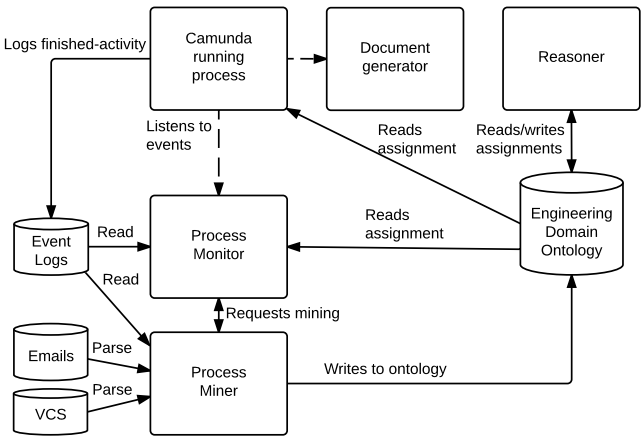

Fig. 9: Software architecture of the prototype

two main databases to store its logs: *i)* a database for processes that are currently executing; and *ii)* a database for historical information. These two databases can be queried through provided Java or REST APIs. Results are returned as either a set of Plain Old Java Objects(POJOs) or in the JSON format, respectively.

Before an activity starts to run, it first fetches the ontology which contains the set of assignments from existing resources to activities. Consecutively, a resource is assigned to the activity and thus can appear on their task list. When the resources complete their tasks, an event is triggered. This event is listened by the process miner and the document generator components, who can react accordingly. At the same time, the event is stored into the Camunda database of the running instances. Both the running processes database and the history database record similarly-structured data. Furthermore, they can be accessed using the same technology, i.e. the Camunda REST APIs. Hence, we abstract both these databases as a single database in Figure 9 and denote it as *Event Logs*.

Reasoner. The reasoner module is implemented as a Java application connected to the Camunda process engine as an asynchronous service. We use Sesame, an open source framework for creating, parsing, storing, inferencing and querying over our ontology data. With respect to the request, the reasoner either performs resource allocation (cf. Figure 10) by first translating the RDF data into the ASP language, solving the problem instance using the ASP solver *clasp*, and then writing the allocation results back to the triple store; or it validates all contained SHACL constraints and returns potential violation result back to the process engine.

Process Monitor. This component is in charge querying the status of the running processes in Camunda. In case a deviation occurs, for example, a process instance cannot be completed within the assigned schedule, the

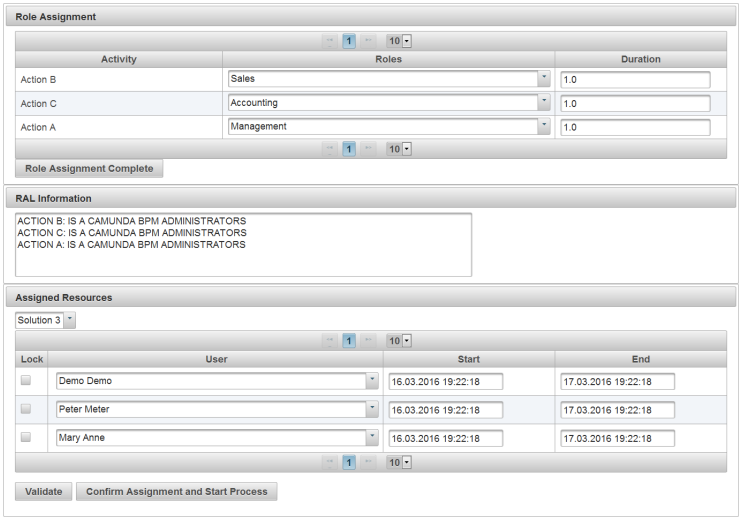

Fig. 10: Resource allocation interface

process monitor must signal out the anomaly. The process adaptation module can use this output to learn the status of the system and subsequently apply an adaptation. This component is implemented as a web client that can read execution logs through the Camunda REST API. Results are returned in the JSON format which are then parsed into POJOs and can be processed by customized monitoring algorithms. In this case the communication happens through periodical queries to the database. An alternative to this is to implement an activity listener that notifies the process monitor whenever a task is completed.

Miner. The miner is in charge of running a number of mining algorithms on the logs from Camunda and from VCSs. Emails and commit messages can also be analysed by using the approaches discussed in [45]. This component is implemented as a web service, which can be called by the process monitor in order to understand how the activities being monitored have performed in the past. Mining algorithms can give new insights into the processes, like for instance actual execution times and several performance indicators of the process. This can contribute to the domain knowledge. Thus, they are stored again into the ontology as RDF.

Document generator. The document generator is in charge of listening to activity submissions and of collecting information from them with the final goal of creating textual documents. This component uses customizable event handlers to process changes of process variables and forms compiled by the users. It is implemented in Java and can be imported as a Java library into several other modules that require document generation from events.

4.1 Limitations

The architecture is currently under implementation. The components have been only individually tested. There is the need for a comprehensive software solution that integrates the single software components into one.

SQL console for querying Camunda logs. We are developing a tool for process monitoring. This tool will allow for SQL-like queries on top of Camunda logs. The approach involves mapping Camundas database schema to RXES [46]. In addition to this we are also developing a tool that can map from RXES to XES [38] and we plan to use this tool with the approach from [47] in order to make it fully compatible with the RXES standard.

Process adaptation. The process adaptation module that we describe in the framework is yet to be implemented. This module will be developed as an intermediate component between the process monitor and the Camunda engine. It will act as a middle layer that is able to correct slight deviations in the running process, without stopping the workflow. Deviations that are not adjustable may occur. In this case, this component will communicate the need for a schedule to the reasoner.

Connection to ontology. Our ontology is currently under improvement. We are planning to complete it with all the data from the engineering domain ontology (cf. Figure 2). Furthermore, its connections to the various components are yet to be implemented.

User interfaces. We support for mining and monitoring techniques whose results are models that are generated out of data. User interfaces to visualize these data are required in order to easy the understanding of the mining results. Analogously, we plan to provide a fully fledged user interface for the reasoner component.

5 Related Work

The existing work on similar frameworks are from safety management [48–50], and decision support domains [51, 52]. To best of our knowledge, there is no framework addressing all the seven requirements (cf. Section 2) that we identify. Therefore, we elaborate on the supporting literature.

Bowen and Stavridou [49] detail the standards concerned with the development of safety-critical systems, and the software in such systems. They identify the challenges of safety-critical computer systems, define the measures for the correctness of such systems and its relevance to several industrial application areas of, e.g. formal methods in railway systems, which is crucial for rigorous and coherent process management.

De Medeiros et al. [52] investigate the core building blocks necessary to enable semantic process mining techniques/tools. They conclude that semantic process mining techniques improve the conventional ones by using three elements, i.e., ontologies, model references from elements in logs/models to concepts in

ontologies, and reasoners. Our framework supports such a high-level semantic analysis through our integrated semantic model and the reasoner module.

Wilke et al. [48] describe a framework for a holistic risk assessment in airport operations. They focus on coordination and cooperation of various actors through a process model derived in BPM, which helps determination of causal factors underlying operation failures, and detection and evaluation of unexpected changes. The holistic consideration of operations handling rules and regulations of their particular domain serves for ensuring compliance. Daramola et al. [50] describe the use of ontologies in a scenario requiring identification of security threats and recommendation of defence actions. Their approach not only help the quick discovery of hidden security threats but also recommend appropriate countermeasures via their semantic framework. By following this approach, they minimize the human effort and enable the formulation of requirements in a consistent way. In our framework we similarly monitor our ontology for compliance checking by querying the ontology via the queries derived from regulations.

Van der Aalst [53] introduced a Petri net based scheduling approach to show that the Petri net formalism can be used to model activities, resources and temporal constraints with non-cyclic processes. Several attempts have also been done to implement the problem as a constraint satisfaction problem. For instance, Senkul and Toroslu [54] developed an architecture to specify resource allocation constraints and a Constraint Programming (CP) approach to schedule a workflow according to the constraints defined for the tasks. Our framework addresses resource allocation via the reasoner module using ASP.

Zahoransky et al. [51] investigate operational resilience of process management. Their approach is proposed as a complementary approach to risk-aware BPM systems, which focuses on detecting the resilience properties of processes based on measures by mining process-logs for decision support to increase process resilience, and therefore provide flexibility. This approach enables agility in run-time and provides a solid foundation for process execution reliability. We address these aspects in our integrated system via the process miner and the process adapter.

6 Conclusions and Future Work

In this paper we have explored challenges of safety-critical human- and data-centric process management in engineering projects which are subject to a large amount of regulations and restrictions, i.e. temporal, resource-related and logistical restrictions, as described in the industry scenario. Our proposed framework addresses all the requirements derived from those challenges upon the general functionality of a BPMS, e.g. process adaptation, resource allocation, document generation and compliance checking.

This work is developed in cooperation with SIEMENS Austria who will be the primary user of the developed system. Our first proof of concept is implemented [55]. Next steps also involve putting in place adaptation mechanisms, implementing and integrating all the components into Camunda, and conducting a thorough evaluation of the implemented system w.r.t. real data from the railway domain.

References

1. G. Fleischanderl, G. E. Friedrich, A. Haselböck, H. Schreiner, and M. Stumptner, "Configuring Large Systems Using Generative Constraint Satisfaction," *IEEE Intelligent Systems*, vol. 13, no. 4, pp. 59–68, 1998.
2. OMG, "BPMN 2.0," recommendation, OMG, 2011.
3. G. Governatori and S. Sadiq, "The Journey to Business Process Compliance," in *Handbook of Research on BPM*, pp. 426–454, IGI Global, 2009.
4. C. Cabanillas, M. Resinas, A. del Río-Ortega, and A. Ruiz-Cortés, "Specification and Automated Design-Time Analysis of the Business Process Human Resource Perspective," *Inf. Syst.*, vol. 52, pp. 55–82, 2015.
5. M. Bozzano and A. Villafiorita, *Design and safety assessment of critical systems*. CRC Press Taylor & Francis Group, 2010.
6. W. van der Aalst, *Process mining: discovery, conformance and enhancement of business processes*. Springer-Verlag Berlin Heidelberg, 2011.
7. M. F. Lopez, A. G. Perez, and N. Juristo, "METHONTOLOGY: from Ontological Art towards Ontological Engineering," in *AAAI97 Symposium*, pp. 33–40, 1997.
8. C. Cabanillas, A. Haselböck, J. Mendling, A. Polleres, S. Sperl, and S. Steyskal, "Engineering Domain Ontology." SHAPE project deliverable.
9. S. Steyskal and A. Polleres, "Defining expressive access policies for linked data using the ODRL ontology 2.0," in *SEMANTICS 2014*, pp. 20–23, 2014.
10. M. C. Suárez-Figueroa, A. Gómez-Pérez, and B. Villazón-Terrazas, "How to write and use the Ontology Requirements Specification Document," in *On the move to meaningful internet systems: OTM 2009*, pp. 966–982, Springer, 2009.
11. N. Russell, W. M. P. van der Aalst, A. H. M. ter Hofstede, and D. Edmond, "Workflow Resource Patterns: Identification, Representation and Tool Support," in *CAiSE*, pp. 216–232, 2005.
12. C. Cabanillas, M. Resinas, J. Mendling, and A. R. Cortés, "Automated team selection and compliance checking in business processes," in *Proceedings of the 2015 International Conference on Software and System Process, ICSSP 2015, Tallinn, Estonia, August 24 - 26, 2015*, pp. 42–51, 2015.
13. C. Cabanillas, M. Resinas, and A. Ruiz-Cortés, "RAL: A High-Level User-Oriented Resource Assignment Language for Business Processes," in *Business Process Management Workshops (BPD'11)*, pp. 50–61, 2011.
14. C. Cabanillas, D. Knuplesch, M. Resinas, M. Reichert, J. Mendling, and A. Ruiz-Cortés, "RALph: A Graphical Notation for Resource Assignments in Business Processes," in *CAiSE*, vol. 9097, pp. 53–68, Springer, 2015.
15. W. Zuberek, "Timed petri nets definitions, properties, and applications," *Microelectronics Reliability*, vol. 31, no. 4, pp. 627–644, 1991.
16. G. Havur, C. Cabanillas, J. Mendling, and A. Polleres, "Automated Resource Allocation in Business Processes with Answer Set Programming," in *BPM Workshops (BPI)*, p. In press, 2015.
17. R. M. Dijkman, M. Dumas, and C. Ouyang, "Semantics and analysis of business process models in BPMN.," *Information & Software Technology*, vol. 50, no. 12, pp. 1281–1294, 2008.
18. R. M. Dijkman, M. Dumas, and C. Ouyang, "Formal semantics and analysis of BPMN process models using Petri nets," Tech. Rep. 7115, Queensland University of Technology, 2007.
19. W. M. Van der Aalst, "The application of petri nets to workflow management," *Journal of circuits, systems, and computers*, vol. 8, no. 01, pp. 21–66, 1998.

20. N. Lohmann, E. Verbeek, and R. Dijkman, "Petri Net Transformations for Business Processes - A Survey," *Transactions on Petri Nets and Other Models of Concurrency II*, vol. 2, pp. 46–63, 2009.
21. M. Weber and E. Kindler, "The petri net markup language.," in *Petri Net Technology for Communication-Based Systems* (H. Ehrig, W. Reisig, G. Rozenberg, and H. Weber, eds.), vol. 2472 of *Lecture Notes in Computer Science*, pp. 124–144, Springer, 2003.
22. L. T. Ly, F. M. Maggi, M. Montali, S. Rinderle-Ma, and W. van der Aalst, "Compliance monitoring in business processes: Functionalities, application, and tool-support," vol. 54, pp. 209–234, March 2015.
23. J. Fuchsbauer, "How to manage Processes according to the European Norm 50126 (EN 50126)." Bachelor thesis, 2015.
24. S. Steyskal, "Engineering Domain Ontology," project deliverable, Siemens, 2016.
25. H. Knublauch and A. Ryman, "Shapes Constraint Language (SHACL)," Working Draft (work in progress), W3C, 2016. `https://www.w3.org/TR/shacl/`.
26. K. R. Bouzidi, C. Faron-Zucker, B. Fies, and N. Le Thanh, "An ontological approach for modeling technical standards for compliance checking," in *Web Reasoning and Rule Systems*, pp. 244–249, Springer, 2011.
27. M. Gebser, R. Kaminski, B. Kaufmann, and T. Schaub, *Answer Set Solving in Practice*. Morgan & Claypool Publishers, 2012.
28. G. Brewka, T. Eiter, and M. Truszczyński, "Answer set programming at a glance," *Communications of the ACM*, vol. 54, no. 12, pp. 92–103, 2011.
29. A. Dovier, A. Formisano, and E. Pontelli, "A comparison of clp (fd) and asp solutions to np-complete problems," in *Logic Programming*, pp. 67–82, Springer, 2005.
30. M. Aschinger, C. Drescher, G. Friedrich, G. Gottlob, P. Jeavons, A. Ryabokon, and E. Thorstensen, "Optimization methods for the partner units problem," in *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pp. 4–19, Springer, 2011.
31. F. Calimeri, M. Gebser, M. Maratea, and F. Ricca, "Design and results of the fifth answer set programming competition," *Artificial Intelligence*, vol. 231, 2016.
32. T. Eiter, G. Ianni, T. Krennwallner, and A. Polleres, "Rules and Ontologies for the Semantic Web," in *Reasoning Web 2008*, vol. 5224, pp. 1–53, 2008.
33. W. M. P. van der Aalst and A. H. M. ter Hofstede, "YAWL: Yet Another Workflow Language," *Inf. Syst.*, vol. 30, no. 4, pp. 245–275, 2005.
34. L. J. R. Stroppi, O. Chiotti, and P. D. Villarreal, "A BPMN 2.0 Extension to Define the Resource Perspective of Business Process Models," in *CIbS'11*, 2011.
35. C. Cabanillas, M. Resinas, J. Mendling, and A. R. Cortés, "Automated team selection and compliance checking in business processes," in *ICSSP*, pp. 42–51, 2015.
36. J. M. A. P. Giray Havur, Cristina Cabanillas, "Resource and data management service architecture." SHAPE project deliverable.
37. B. F. Van Dongen, A. K. A. De Medeiros, H. M. W. Verbeek, A. J. M. M. Weijters, and W. M. P. Van Der Aalst, "The ProM framework: A new era in process mining tool support," in *Lect. Notes Comput. Sci.*, vol. 3536, pp. 444–454, Springer, 2005.
38. H. M. W. Verbeek, J. C. A. M. Buijs, B. F. Van Dongen, and W. M. P. Van Der Aalst, "XES, XESame, and ProM 6," in *Lect. Notes Bus. Inf. Process.*, vol. 72 LNBIP, pp. 60–75, Springer, 2011.
39. E. Kindler, V. Rubin, and W. Schäfer, "Activity Mining for Discovering Software Process Models," *Softw. Eng.*, vol. 79, pp. 175–180, 2006.

40. W. Poncin, A. Serebrenik, and M. V. D. Brand, "Process Mining Software Repositories," *2011 15th Eur. Conf. Softw. Maint. Reengineering*, pp. 5–14, 2011.
41. C. Di Ciccio, M. Mecella, M. Scannapieco, D. Zardetto, and T. Catarci, "MailOfMine - Analyzing mail messages for mining artful collaborative processes," *Lect. Notes Bus. Inf. Process.*, vol. 116 LNBIP, pp. 55–81, 2012.
42. S. Bala, C. Cabanillas, J. Mendling, A. Rogge-Solti, and A. Polleres, "Mining Project-Oriented Business Processes," in *BPM*, pp. 425–440, 2015.
43. C. M. Pilato, B. Collins-Sussman, and B. W. Fitzpatrick, *Version control with subversion*. " O'Reilly Media, Inc.", 2008.
44. L. Torvalds and J. Hamano, "Git: Fast version control system," *URL http//git-scm. com*, 2010.
45. C. Cabanillas, S. Bala, J. Mendling, and A. Polleres, "Combined method for mining and extracting processes, related events and compliance rules from unstructured data," tech. rep., WU Vienna, 2016.
46. B. F. van Dongen and S. Shabani, "Relational XES : Data Management for Process Mining," *BPM Cent. Rep. BPM-15-02*, 2015.
47. S. Schönig, C. Cabanillas, S. Jablonski, and J. Mendling, "Mining the Organisational Perspective in Agile Business Processes," in *BPMDS*, pp. 37–52, 2015.
48. S. Wilke, A. Majumdar, and W. Y. Ochieng, "Airport surface operations: A holistic framework for operations modeling and risk management," *Safety Science*, vol. 63, pp. 18–33, 2014.
49. J. Bowen and V. Stavridou, "Safety-critical systems, formal methods and standards," *Software Engineering Journal*, vol. 8, no. 4, pp. 189–209, 1993.
50. O. Daramola, G. Sindre, and T. Moser, "A tool-based semantic framework for security requirements specification.," *J. UCS*, vol. 19, no. 13, pp. 1940–1962, 2013.
51. R. M. Zahoransky, C. Brenig, and T. Koslowski, "Towards a process-centered resilience framework," in *Availability, Reliability and Security (ARES), 2015 10th International Conference on*, pp. 266–273, IEEE, 2015.
52. A. K. A. de Medeiros, W. Van der Aalst, and C. Pedrinaci, "Semantic process mining tools: core building blocks," 2008.
53. W. van der Aalst, "Petri net based scheduling," *Operations-Research-Spektrum*, vol. 18, no. 4, pp. 219–229, 1996.
54. P. Senkul and I. H. Toroslu, "An Architecture for Workflow Scheduling Under Resource Allocation Constraints," *Inf. Syst.*, vol. 30, pp. 399–422, July 2005.
55. S. Bala, G. Havur, S. Sperl, S. Steyskal, A. Haselböck, J. Mendling, and A. Polleres, "SHAPEworks: A BPMS Extension for Complex Process Management," in *BPM Demos*, To appear, 2016.