Evaluation report (version 1 – input for refinements)

Deliverable D6.3

FFG – IKT der Zukunft
SHAPE Project
2014 – 845638

Project acronym:	SHAPE
Project full title:	Safety-critical Human- & dAta-centric Process management in Engineering projects
Work package: Document number: Document title: Version:	6 6.3 Evaluation report (version 1 – input for refinements) 1
Delivery date: Actual publication date: Dissemination level: Nature:	October, 1st, 2016 (M4) - Public Report
Editors / lead beneficiary : Author: Reviewers:	SIEMENS AG Srdjan Stevanetic Alois Haselböck and Jan Mendling

Table 1. Document Information

Contents

1. Introduction

In SHAPE, we aim at developing 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.

The goal of this document is to evaluate the existing system's architecture, i.e. to provide information on how the system's architecture complies with the specified requirements. With regard to that, we first describe the architecture and then provide the corresponding evaluation. Regarding the evaluation part, we provide an initial, qualitative evaluation of the architecture by evaluating the current system state with regard to the system functional and non-functional requirements and indicating the potential risks related to the current and new implementation.

This document is organized as follows. Section 2 describes the main application scenario. Section 3 describes both modeling time and runtime requirements of the system. Section 4 describes the generic architecture of our framework for process management in complex engineering projects while Section 5 describes the concrete prototype architecture. Section 6 describes the evaluation of the current prototype architecture with regard to both functional and non-functional requirements. Section 7 provides conclusions that wrap up the given evaluation points. In Section 8, we provide a list of references used in the document.

2. 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. Figure 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) [1]. 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 grey 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 has a great importance in large-scale engineering processes in which unexpected situations may have critical consequences, e.g. delays that lead to unplanned higher costs. Many resources are involved in the engineering process, ranging from laboratories and specific hardware to the employees of the organization, who are responsible for the correct execution of the process.

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 sub-process *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 S1 and the test execution for a system S2, another engineer who cannot be involved in the testing of system S1 but can perform the two activities for the system S2, 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.

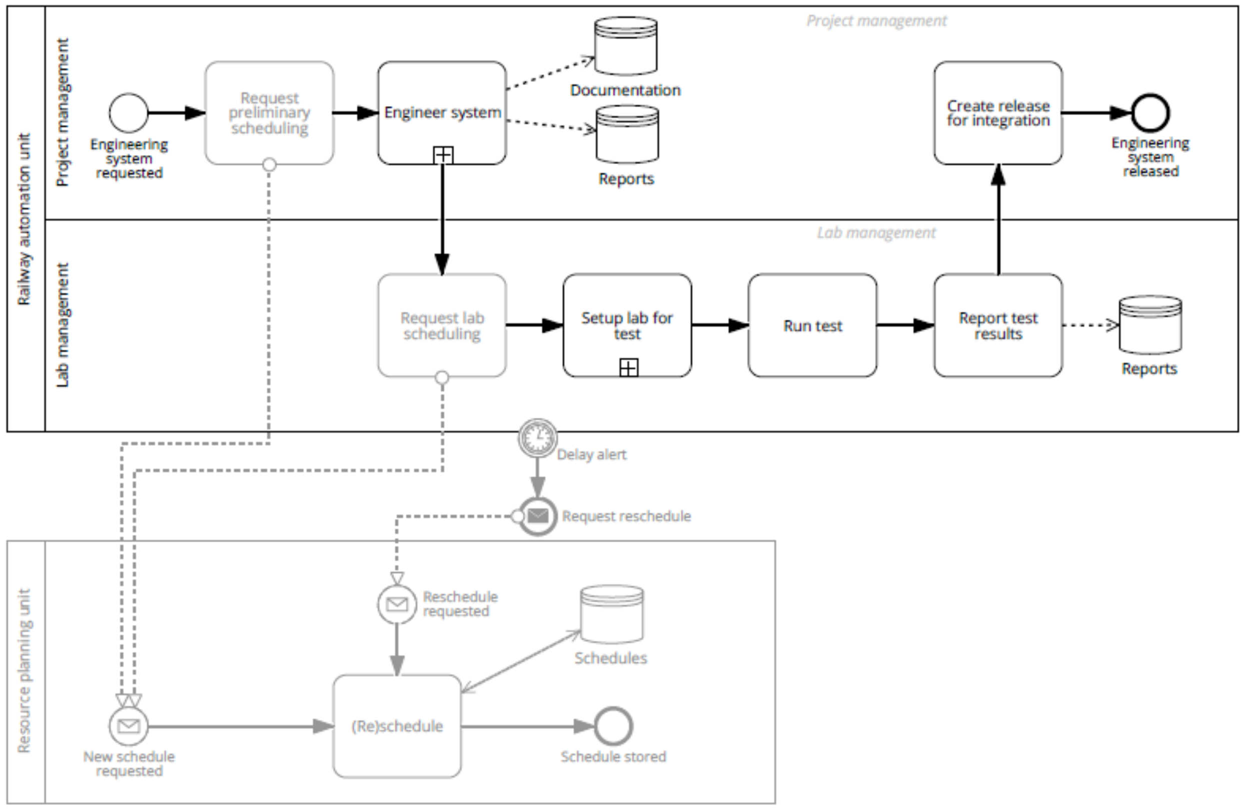

Figure 1. Engineering process for building a new infrastructure system in the railway automation domain

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. 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.

3. Requirements

Based on the provided industry scenario the following set of requirements is identified to be relevant for the SHAPE project: Table 2 lists the modeling time while Table 3 lists the runtime system requirements.

Requirement	Description
RM01	Automatic extraction of process models from textual descriptions
RM02	Automatic extraction of process models from log data (process mining)
RM03	RA can model their processes / change their process models without help of R&D
RM04	Automatic generation of process documentation (process handbook)
RM05	Definition of roles
RM06	Access rights of roles modelling
RM07	Definition of constraints to check data integrity, e.g., file versions, tool versions
RM08	Definition of constraints to check data completeness, e.g. do ReleaseNotes exist?
RM09	Explicit modelling of safety aspects, e.g., 4eye principle, engineer != verifier
RM10	Safety risks analysis on processes, e.g., for support of hazard analysis
RM11	Task duration and deadline modelling
RM12	Escalation modelling
RM13	Specification of templates (documents, e-mails, checklists) with placeholders for artefacts, e.g., e-mail template for release letter of tool outputs to subsequent engineers, e.g., release letter to the customer, with actual data file links/locations/versions, e.g., checklist template for verifiers

Table 2. SHAPE modeling time requirements

Requirement	Description
RR01	Process engine monitors/guides engineers at runtime
RR02	Notification/requests of process steps via e-mail
RR03	Notification/requests of process steps via web interface
RR04	Complete documentation of process steps during runtime (log file)
RR05	Integration/examination of e-mail traffic, e.g., tags in emails cause starting/finishing of activities
RR06	Integration/examination of SharePoint traffic, e.g., finished/changed documents start/finish activities
RR07	Integration/examination of SVN/GIT traffic, e.g., commit automatically finishes activity
RR08	Integration of people directory
RR09	Assignment of actual persons to roles (people resolution)
RR10	Assignment of actual persons to tasks based on availability/workload/custom weighting
RR11	Access rights checking (roles based, based on policy model)
RR12	Deadline and effort tracking by logging of activity/process durations
RR13	Deadline and effort prediction, e.g., not enough available engineer-hours to meet deadline
RR14	Synchronization with project planning, e.g., with Enterprise Project Management (EPM)
RR15	Detection of process deviations e.g., incomplete input data used
RR16	Predictive process deviation, e.g., a predicted failed deadline causes creation of new activities like a meeting

RR17	Rollback or safe process continuation in case of a deviation, e.g., wrong input data -> use correct input data
RR18	Compensation handling
RR19	Continuous process optimization (automatic detection of process improvements)
RR20	Learning of task durations / variations
RR21	Automatic generation of documents/e-mails/checklists – based on templates cf. RM14
RR22	Automatic generation of a plan inventory, i.e. a document consisting of all relevant artefacts, their locations and versions
RR23	Checking of data integrity and completeness constraints and user notification of constraint violations cf. RM08, RM09

Table 3. SHAPE runtime requirements

4. Framework for Process Management in Complex Engineering Projects

Typical functionality of a BPMS includes modeling and executing processes. Information about process instances is usually stored in event logs including, among others, temporal and resource information related to the execution of the process activities [2]. In addition to that structured information, several kinds of unstructured and semi-structured 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, e.g., deviations with regard to the expected behavior. The *Process Miner* component of our framework (see Figure 2) tries to discover as much data relevant to the current state of a process execution as possible, performs the transformations required as specified by, and communicates the information extracted to the *Process Monitor* 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. 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. We encode the resource allocation problem in Answer Set Programming (ASP) [3], 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. Finally, the Document Generator of the framework provides support for by helping to fill out the documents that must be generated as output of process activities. As aforementioned, this automation is expected to decrease reporting errors, especially in documents related to auditing.

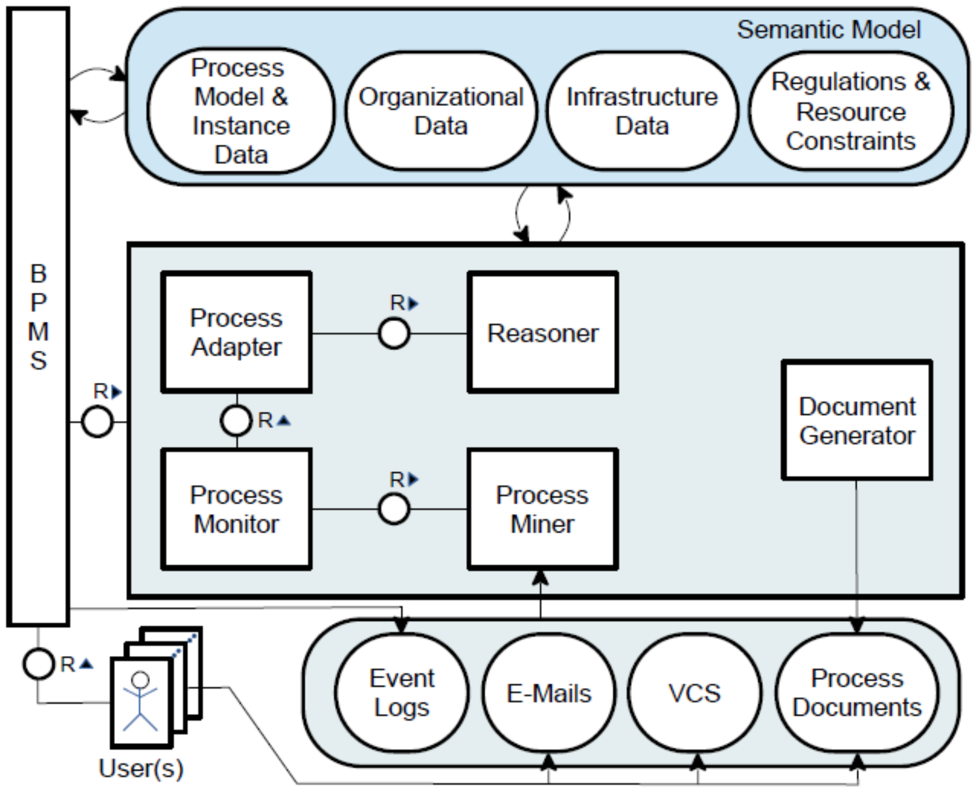

Figure 2. Proposed framework for process management in complex engineering projects

Aiming at automation, we believe that an ontology is the most appropriate mechanism for storing and retrieving data due to, among others, the large amount of off-the-shelf reasoners available to query them. Therefore, following the METHONTOLOGY approach [4], we have developed an engineering domain ontology that represents: (i) engineering domain and organizational (i.e. resource-related) knowledge; (ii) business processes; and (iii) regulations and policies [5].

Regarding the engineering domain and organizational knowledge, we decided to adopt parts of the organizational meta-model described in [6] and enriched it with concepts for modeling teams [7] (see Figure 3).

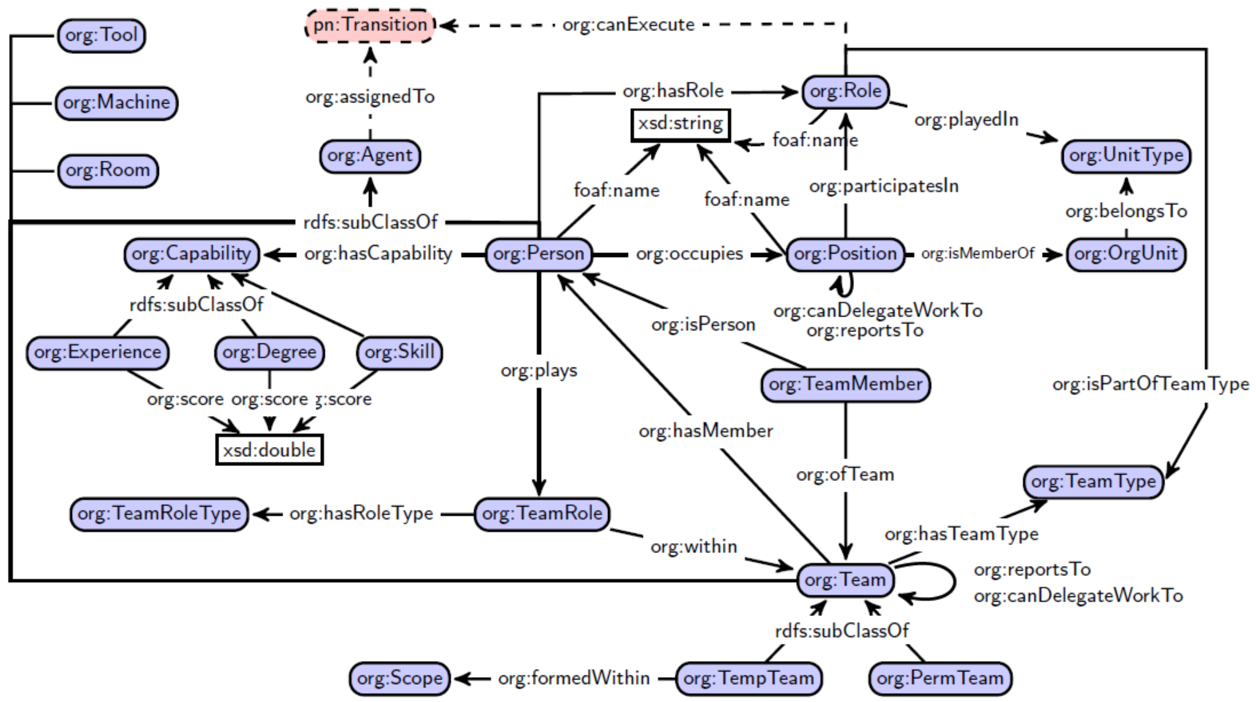

Figure 3. Ontology for engineering domain and organizational knowledge

Regarding the business processes ontology, we decided to represent processes and process instances using timed Petri nets utilizing transformation rules proposed in [8, 9] (see Figure 4).

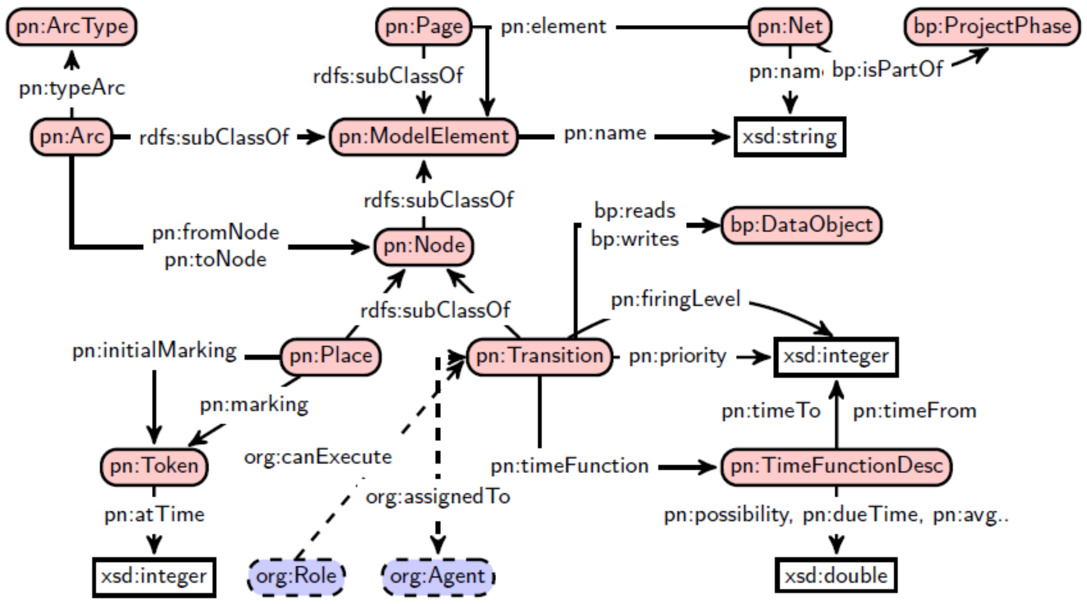

Figure 4. Ontology for processes and process instances

Extracting and specifying compliance rules is 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 regulations and policies such as domain-specific norms or workflow patterns. Since all process relevant data is stored in RDF, we plan to utilize recent advancements in the area of constraint checking for RDF, i.e. the Shapes Constraint Language (SHACL) [10] for representing and validating identified compliance rules. Specified compliance rules and constraints are then subsequently used by a monitoring/compliance checking engine for verifying correct and valid execution of business processes.

5. Prototype Architecture

Here, we coarsely describe main implementation details of the architecture prototype components and their interactions. The prototype architecture is shown in Figure 5.

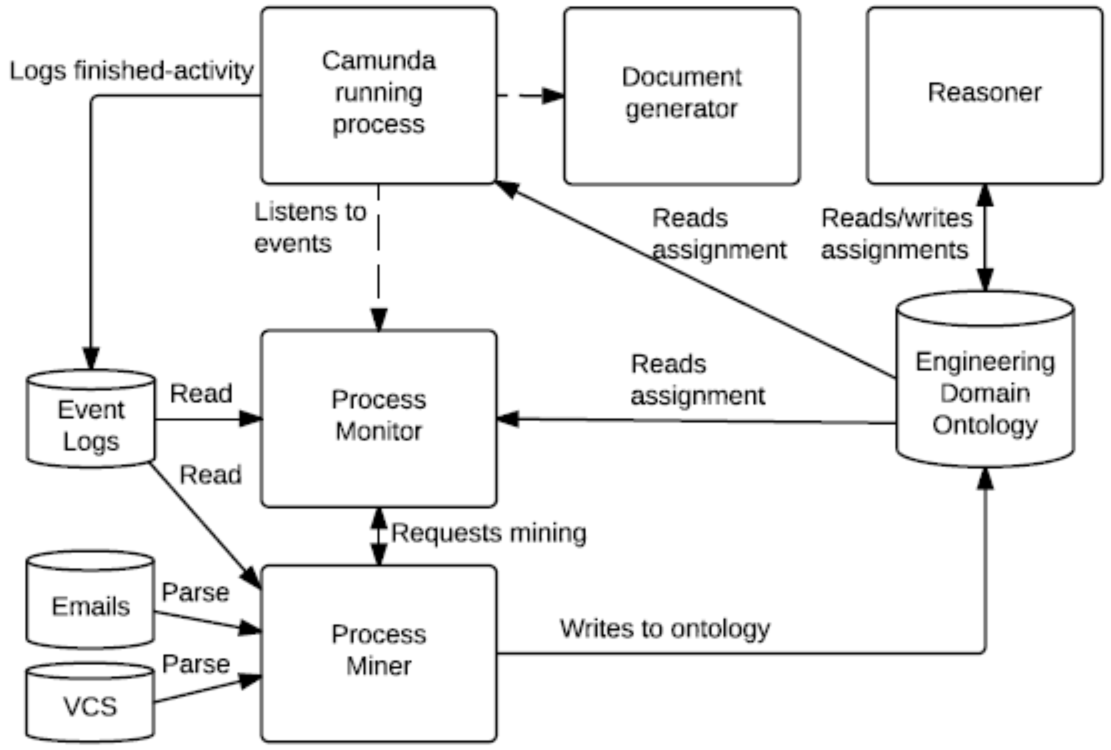

Figure 5. Prototype architecture

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 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.

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 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 or it validates all contained constraints specified in the ontology 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 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 [11]. 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.

6. Initial Architecture Review

In this part, we pursue the initial, qualitative review of the described system's architecture. Based on the examined technical design of the architecture and its documentation, we evaluate the current system state with regard to the system requirements. The evaluation of both functional and non-functional requirements is provided. They show how the given requirements are satisfied or how they can be satisfied as well as which potential risks are associated with them.

Tables 4 and 5 show the evaluation of the modeling time and runtime requirements from Tables 2 and 3, respectively. For the sake of easier tracking, we list again the set of requirements with their descriptions (Columns 1 and 2 in the tables).

Requirement	Description	Evaluation with regard to the current architecture
RM01	Automatic extraction of process models from textual descriptions	To be implemented (some approaches for mining processes from unstructured data exist in the literature)
RM02	Automatic extraction of process models from log data (process mining)	To be implemented (traditional process mining algorithms)
RM03	RA can model their processes / change their process models without help of R&D	The users can change process models using the Camunda UI, but need to know how to deploy the changes on the server in case that it is required
RM04	Automatic generation of process documentation (process handbook)	.doc files are generated by the framework. Any other format generation can be implemented using the delegation technique that Camunda supports (e.g. JavaDelegate)
RM05	Definition of roles	Defined as a part of the ontology. The ontology needs to be synchronized with the definitions supported by Camunda (e.g. roles definitions using Camunda Admin Web App.)
RM06	Access rights of roles modelling	The same as for RM05.
RM07	Definition of constraints to check data integrity, e.g., file versions, tool versions	Can be specified using the delegation technique in Camunda. In the future, it needs to be synchronized with the ontology, because the ontology should contain and manage all kinds of constraints
RM08	Definition of constraints to check data completeness, e.g. do ReleaseNotes exist?	The same as for RM05.
RM09	Explicit modelling of safety aspects, e.g., 4eye principle, engineer != verifier	Safety aspects can be modelled in the ontology as constraints (to be implemented). User can edit the ontology using appropriate ontology editor that facilitates the definition of the safety aspects.
RM10	Safety risks analysis on processes, e.g., for support of hazard analysis	To be implemented in the Process Monitor component.
RM11	Task duration and deadline modelling	This is supported by the scheduling part of the system (Reasoner component). The introduced Answer Set Programming solver, used for the resource allocation, scales bad with size so that other logic might need to be implemented. Another scheduling that could be implemented is stochastic scheduling, i.e. the assessment of the resources assignments using statistical process execution
RM12	Escalation modelling	Escalation modelling is managed by the exception handler that already exists in the BPMN standard of Camunda. Therefore the exceptions can be appropriately handled.
RM13	Specification of	In the document generator, .doc files comments can be

	templates (documents, e-mails, checklists) with placeholders for artefacts e.g., e-mail template for release letter of tool outputs to subsequent engineers, e.g., release letter to the customer, with actual data file links/locations/versions, e.g., checklist template for verifiers	added to the document in order to specify which information is to be documented (e.g. project name, users, tasks). Email and other kinds of templates need to be implemented.

Table 4. Evaluation of the modeling time requirements with regard to the current system's architecture

Requirement	Description	Evaluation with regard to the current architecture
RR01	Process engine monitors/guides engineers at runtime	All activities in the process are inherently stored in Camunda logs.
RR02	Notification/requests of process steps via e-mail	To be implemented. It can be added using Camunda delegation technique.
RR03	Notification/requests of process steps via web interface	Inherently implemented via task lists in Camunda
RR04	Complete documentation of process steps during runtime (log file)	The same as for RR01.
RR05	Integration/examination of e-mail traffic, e.g., tags in emails cause starting/finishing of activities	To be implemented. A problem that might occur is a security problem during accessing of the companies' servers infrastructure
RR06	Integration/examination of SharePoint traffic, e.g. finished/changed documents start/finish activities	To be implemented as a part of the process mining.
RR07	Integration/examination of SVN/GIT traffic e.g., commit automatically finishes activity	SVN analysis is implemented.
RR08	Integration of people directory	To be implemented.
RR09	Assignment of actual persons to roles (people resolution)	To be defined in the ontology constraints or optimized by the Reasoner component using the ASP logic.
RR10	Assignment of actual persons to tasks based on availability/workload/custom weighting	Implemented by scheduling.
RR11	Access rights checking (roles based, based on policy model)	Implemented in the Ontology constraints.
RR12	Deadline and effort tracking by logging of activity/process durations	The same as for RR01.
RR13	Deadline and effort prediction e.g., not enough available engineer-hours to meet deadline	Implemented in scheduling.
RR14	Synchronization with project planning e.g., with Enterprise Project Management (EPM)	To be implemented. The calculated scheduled distribution of assignments can be aligned with the project plan, i.e. the requirements from the project plan can be added as constraints to the scheduling process.
RR15	Detection of process deviations e.g., incomplete input data used	Inherently implemented in the Camunda validation, e.g. when the task is completed validation event is triggered.
RR16	Predictive process deviation e.g., a predicted failed deadline causes creation of new activities like a meeting	To be implemented.

RR17	Rollback or safe process continuation in case of a deviation e.g., wrong input data -> use correct input data	Inherently provided by Camunda, e.g. transactions management, i.e. rolling back to the previous stable state.
RR18	Compensation handling	To be implemented. The main focus of compensation is to set back the data and state consistency where an automatic transactional rollback is not available. It is very useful technique for long running business processes.
RR19	Continuous process optimization (automatic detection of process improvements)	To be implemented by mining the process logs and adapting the given process. Implementing adaptations can require complex changes in the process that sometimes would require creating a new instance of a process. In that case, the new instance has to be backward compatible with the previous one.
RR20	Learning of task durations / variations	The same as for RR01.
RR21	Automatic generation of documents/e-mails/checklists – based on templates cf. RM14	Implemented in the document generation.
RR22	Automatic generation of a plan inventory, i.e. a document consisting of all relevant artefacts, their locations and versions	The same as for RR21.
RR23	Checking of data integrity and completeness constraints and user notification of constraint violations cf. RM08, RM09	Constraints are defined in the ontology and validated in the Reasoner component.

Table 5. Evaluation of the runtime requirements with regard to the current system's architecture

In Tables 4 and 5, we provide the evaluation of the functional requirements of the system. The evaluation of the non-functional requirements (quality attributes) is shown Table 6.

Quality attribute	Description	Evaluation with regard to the current architecture
Flexibility	Flexibility reflects the ease with which a system or component can be modified for use in applications or environments other than those for which it was specifically designed.	Easy changes: document generation, scheduling analysis using the ASP logic, ontology modifications with regard to the domain Difficult changes: Camunda process engine integration with another process engine (risk: incompatible process integration APIs)
Extensibility	Extensibility is a software design principle defined as a system's ability to have new functionality extended, in which the system's internal structure and data flow are minimally or not affected, particularly that recompiling or changing the original source code is unnecessary when changing a system's behavior, either by the creator or other programmers.	Scripting is inherently provided by Camunda (e.g. .js, .groovy external scripts can be called). In that way any additional operations with tasks can be easily integrated. Multiple process definitions (versions) can exist in parallel and communicate with each other.
Interoperability	Interoperability is the ability of a system or different systems to operate successfully by communicating and exchanging information with other external systems written and run by external parties. An interoperable system makes it easier to exchange and reuse information internally as well as externally.	Camunda supports REST and Java APIs that enable monitoring and interaction with the process engine. The communication with other external systems like SVN or emails can be easily integrated by using the Camunda delegation technique.
Rausability	Reusability defines the capability for components and subsystems to be suitable for use in other applications and in other scenarios. Reusability minimizes the duplication of components and also the implementation time.	Scheduling and document generation components are independent from Camunda and can be reused in any related scenario. Camunda process engine limits the reusability in terms of process models integration and interoperability. Ontology specification can be reused.
Complexity	Architectural complexity reflects the use of code, components, architectural styles, best practices, design patterns, etc. beyond the minimum needed to fulfill the functional requirements to the system.	The system is pretty complex and use different technologies: java ee technology stack, Camunda, RDF ontology. This can complicate testing.

	Architectural complexity can be measured in terms of code size, number of components and classes, and entanglement (i.e., lack of separation of concerns). Architectural complexity is not necessarily induced by programming complexity, as defined in software engineering literature (e.g., cyclomatic complexity). Rather, it is a measure of how much effort and money has to be spent at present and in the future for keeping the system running and fulfilling its function.	
Safety	Safety is concerned that especially life critical systems behave as required (doing no or minimal harm to other systems/devices) even when components fail. Safety requirements are the shall not requirements which exclude unsafe situations from the possible solution space of the system. The capabiltity of the software product to achieve acceptable levels of risk of harm to people business software property or the environment in a specified context of use. The subtype of defensibility that is the degree to which the system or architectural component prevents or reduces the probability or severity of detects and properly reacts to Unauthorized unitentional harm to valuable assets Mishaps Hazards Safety risks	Camunda transactions inherently support safe execution of processes (risk: one should know how to specify Camunda transactions which is domain specific as well as how to safely undo external/non-Camunda actions, e.g. send an ignore email mail) Safety constraints for the process can be defined/verified in the ontology (risk of potential unsatisfiable constraints)
Maintainability	Maintainability is the ability of the system to undergo changes with a degree of ease. These changes could impact components, services, features, and interfaces when adding or changing the functionality, fixing errors, and meeting new business requirements.	See the Complexity quality attribute.
Composability	Composability is a system design principle that deals with the interrelationships of components. A highly composable system provides recombinant components that can be selected and assembled in various combinations to satisfy specific user requirements.	Camunda can be composed with other programs using its REST APIs (risk: rest api do not provide full ability of controlling the process engine like java APIs that are Camunda specific).
Auditability	Auditability is the degree to which transactions can be traced and audited through a system. Auditability means that: (1) it is possible to establish whether a system is functioning properly and,	To be defined in the ontology constraints or optimized by the Reasoner component using the ASP logic.

	thereafter, that it has worked properly, (2) the capability of supporting a systematic, independent and documented process for obtaining audit evidence and evaluating it objectively to determine the extent to which audit criteria are fulfilled.	
Reliability	Reliability is the ability of a system or component to perform its required functions under stated conditions for a specified period of time In general reliability is the ability of a person or system to perform and maintain its functions in routine circumstances as well as hostile or unexpected circumstances.	ASP scheduling logic can sometimes consume too many resources in order to find a reliable solution and it can be a potential risk in terms of the system response time.

Table 6. Evaluation of the non-functional requirements with regard to the current system's architecture

Additional evaluation points:

- The documentation of the system needs to be improved: different system views can be generated in order to facilitate the understanding of the system (e.g. typical 4 architectural views logical, process, development, and deployment). Some partial description of these views already exists (i.e. the component and deployment diagrams)
- To facilitate the understanding of the mining, monitoring, and scheduling components, user interfaces for visualizing these data would be useful.
- Connections among different components in the system have to be specified using corresponding interfaces. For example, connecting ontology with other components in an appropriate way need to be implemented.

7. Conclusion

To summarize the pursued architecture evaluation we can say the following. Regarding the required system functionalities, some parts still need to be implemented. However, the necessary infrastructure for successfully implementing those parts already exists. Regarding the system non-functional requirements, most of them are or can be successfully satisfied by further improvements. However, satisfying some non-functional requirements is very difficult because of the specificities inherently contained in the chosen software technologies (e.g. the Camunda process engine). The system as a whole is pretty complex, dealing with many different technologies, which can be a potential bottleneck for the testing and maintenance. To facilitate the understanding of the system, the system documentation needs to be improved, the interfaces between the components need to be clearly defined, and the appropriate UIs need to be provided.

8. References

1. OMG, \BPMN 2.0," recommendation, OMG, 2011.
2. Wil M. P. van der Aalst. 2011. *Process Mining: Discovery, Conformance and Enhancement of Business Processes* (1st ed.). Springer Publishing Company, Incorporated.
3. 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.
4. M. Bozzano and A. Villa_orita, Design and safety assessment of critical systems. CRC Press Taylor & Francis Group, 2010.
5. M. Bozzano and A. Villa_orita, Design and safety assessment of critical systems. CRC Press Taylor & Francis Group, 2010.
6. Anonymous, \Details omitted for double-blind reviewing," 2015.
7. S. Steyskal and A. Polleres, \De_ning expressive access policies for linked data using the ODRL ontology 2.0," in SEMANTICS 2014, pp. 20-23, 2014.
8. M. C. Suarez-Figueroa, A. Gomez-Perez, and B. Villazon-Terrazas, "How to write and use the Ontology Requirements Speci_cation Document," in On the move to meaningful internet systems: OTM 2009, pp. 966{982, Springer, 2009.
9. N. Russell, W. M. P. van der Aalst, A. H. M. ter Hofstede, and D. Edmond, "Workow Resource Patterns: Identi_cation, Representation and Tool Support," in CAiSE, pp. 216{232, 2005.
10. C. Cabanillas, D. Knuplesch, M. Resinas, M. Reichert, J. Mendling, and A. Ruiz-Cortes, "RALph: A Graphical Notation for Resource Assignments in Business Processes," in CAiSE, vol. 9097, pp. 53-68, Springer, 2015.
11. F. Calimeri, M. Gebser, M. Maratea, and F. Ricca, "Design and results of the fifth answer set programming competition," Articial Intelligence, vol. 231, 2016.