Facilitating the Compliance of Process Models with Critical System Engineering Standards using Natural Language Processing

Faiz Ul Muram[1][a], Muhammad Atif Javed[2][b] and Samina Kanwal[3][c]

[1]*School of Innovation, Design and Engineering, Mälardalen University, Västerås, Sweden*

[2]*RISE, Research Institutes of Sweden, Västerås, Sweden*

[3]*National University of Sciences and Technology, Islamabad, Pakistan*

Keywords: Process Engineering, Standard Documents, Natural Language Processing, Mapping Rules, Model Extraction, SPEM 2.0, EPF Composer, Compliance Management.

Abstract: Compliance of process models with relevant standards is mandatory for certifying the critical systems. However, it is often carried out in a manual manner, which is complex and labour-intensive. Previous studies have not considered the automated processing of standard documents for achieving and demonstrating the process compliance. This paper leverages natural language processing for extracting the normative process models embedded in the standard documents. The mapping rules are established for structuring the standard requirements and content elements of process models, such as tasks, roles and work products. They are organized into a process structure by considering the phases, activities and milestones. During the planning phase, the standard requirements, process models and compliance mappings are generated in EPF Composer; it supports the major parts of the OMG's Software & Systems Process Engineering Metamodel (SPEM) 2.0. The reverse compliance of extended or pre-existing process models can be carried out during the execution phase; specifically, the compliance gaps are detected, possible measures for their resolution are provided and missing elements are added after the process engineer approval. The applicability of the proposed methodology is demonstrated for the ECSS-E-ST-40C compliant space system engineering process.

1 INTRODUCTION

Compliance of process models with relevant standards is mandatory for getting the approval or acceptance from the certification body. The normative process models embedded in the standards typically include planning of units of work such as activities and tasks, which are expected to be executed during the development; work products to be taken as input or produced as output; involved roles, and a set of methods to be used. To achieve and demonstrate process compliance, the standard documents are processed. In addition to the extraction of standard requirements and process, there is a need to provide the reference between them. However, it is a complex and labour-intensive task to extract, understand, model such knowledge and manage traceability in a manual manner (Javed et al., 2016; Javed et al., 2018). The published studies focused on the automatic extraction from natural language descriptions and mapping to the Business Process Model and Notation (BPMN) (OMG, 2011). To date, however, the automatic extraction from standard documents and generation of Software & Systems Process Engineering Metamodel (SPEM) (OMG, 2008) compliant requirements, process models and compliance mappings have not been considered. Both SPEM and BPMN are Object Management Group (OMG) specifications. SPEM 2.0 provides additional information structures needed for the processes modelled with UML 2.0 activities or BPMN to describe an actual development process (OMG, 2008).

This paper focuses on the compliance management during planning and execution phases. The standard documents are processed for achieving and demonstrating the process compliance. The extracted parts of standard documents are mapped to the requirements and process elements based on the specified rules. The requirements hierarchy is maintained, while the reusable content elements, such as tasks, work products, roles, and guidance are organized in the process structure, with the consideration of iteration, phase, activity and milestone elements. During the planning phase, the SPEM-compliant require-

[a] https://orcid.org/0000-0001-6613-4149
[b] https://orcid.org/0000-0003-3781-4756
[c] https://orcid.org/0000-0002-1079-9712

Muram, F., Javed, M. and Kanwal, S.
Facilitating the Compliance of Process Models with Critical System Engineering Standards using Natural Language Processing.
DOI: 10.5220/0010455903060313
In *Proceedings of the 16th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2021)*, pages 306-313
ISBN: 978-989-758-508-1

Table 1: Process Modelling Elements in SPEM.

Task	Role	Work Product	Category	Domain	Discipline	Tool Category	Role Set
Guidance	**Practice**	**Activity**	**Capability Pattern**	**Delivery Process**	**Iteration**	**Phase**	**Milestone**

ments, process models and compliance mappings are generated in its reference implementation, more specifically the Eclipse Process Framework (EPF) Composer[1] is utilised. By automatically generating requirements and process models within an environment that supports traceability, this paper facilitates the compliance management and demonstration during the planning phase. The reverse compliance of process during the execution phase is also supported. Specifically, we provide comprehensive feedback to inform the compliance gaps, possible options for their resolutions, and take measures based on process engineer feedback. The applicability of the proposed methodology is demonstrated by extracting text from the space engineering standard documents, generating the requirements, process models and compliance mappings in EPF Composer, and reverse compliance of process models with standards.

The rest of this paper is organized as follows: Section 2 discusses the process engineering metamodel. Section 3 presents the methodology for automatic extraction of requirements and process elements, their organization and mappings, generation of corresponding models in EPF Composer, as well as the reverse compliance with standards. Section 4 demonstrates the applicability of proposed methodology for ECSS-E-ST-40C standard. Section 5 discusses the related work. Section 6 concludes the paper and presents future research directions.

2 PROCESS ENGINEERING METAMODEL

SPEM 2.0 (OMG, 2008) is the OMG's standard, which provides the necessary concepts for modelling, documenting, interchanging, and managing systems and software development processes. The framework of SPEM 2.0 comprises of *Method Content* and the *Process*. Method Content provides support for the definition of reusable process content, i.e., *Task*, which defines the unit of work being performed by one or many *Role(s)* and may generate the *Work Products* that can be a type of artefact, deliverable, or outcome; and the *Category* that can be used to categorize any number of content elements, such as *Domain*, *Discipline*, *Tool Category* and *Role Set*. *Guidance* is defined in the intersection of Method Content and Process. It describes additional information of work and is classified into various kinds such as guideline, practice, example, etc. *Process* takes the content elements and relate them into partially-ordered sequence of work in breakdown structures that are customized to specific types of projects. For instance, *Phase* that represents a significant period in a project and normally ends with major milestones, or a set of deliverables. To define a *Process*, tasks can be grouped to form an *Activity* and a set of nested activities can be grouped into *Iteration* in order to indicate that the set can be repeated more than once. There are two process kinds: (1) process patterns (referred as capability patterns in EPF Composer) are building blocks that hold process knowledge for a key area of interest; and (2) complete life cycles that are modelled as delivery processes. Table 1 shows the main structural elements for defining the process.

EPF Composer is an extensible process engineering framework, based on the OMG's SPEM. It is evolved from Eclipse Galileo 3.5.2 to Eclipse Neon 4.6.3 (Javed and Gallina, 2018; Javed et al., 2019) after 11 years. It is possible to launch the EPF Composer as a stand-alone application, but also in the Eclipse integrated development environment (IDE). EPF Composer is used to model standards and development process, as well as to show that process comply with standards (Muram et al., 2018; Muram et al., 2019). In EPF Composer, a method library is a repository of method elements, which is composed of a set of *Method Plugins* and *Configurations*. The *Method Plugins* are containers of process related information, while a *Configuration* is a selection of sub-sets of library content to be shown in the browsing perspective. To model the requirements listed in the standards, the guidance kinds *Practice* can be cus-

[1]https://www.eclipse.org/epf/

tomized with an icon in a separate *Customized_Icon* plugin. The *Standards_Requirements* plugin captures the standard's requirements and has the variability relationship *Extends* with the previously mentioned plugin. The *Process_Lifecycle* plugin defines the process life cycle, whereas the mapping between standard requirements and process life cycle is provided in *Mapping_Requirements* plugin (Muram et al., 2018; Muram et al., 2019).

3 METHODOLOGY

This section describes our proposed methodology to facilitate the approval process, more specifically, the automatic extraction, generation and management of process-related compliance information. The natural language processing of standard documents is carried out for automatic extraction of requirements and process models (see Section 3.1). The mapping rules are defined for structuring of extracted information in compliant with SPEM 2.0 (see Section 3.2). During the planning phase, the requirements, process models and compliance mappings are generated (see Section 3.3). After that, the engineers can update or evolve the generated models and provide the evidence of compliance for review. The reverse compliance of process models is also supported during the execution phase (see Section 3.4). The emphasis is placed on avoiding re-generation and replacement of extended models. Accordingly, the support is provided for identifying gaps with standards, determining possible measures for their resolution, and updating parts of models based on process engineer feedback. The process compliance could be further explained via argumentation (Muram et al., 2018).

3.1 Natural Language Processing

The standards are typically available in Word or Portable Document Format (PDF). In some cases, the Excel file containing all standard requirements is provided; the standard documents may need to be processed in conjunction. Their natural language processing includes tokenization, part-of-speech tagging and named entity recognition. The part containing standard requirements is located. In addition, the clauses, part-based distinction criteria and hierarchical levels in standard documents are considered. The standard documents may contain introductory material, examples, note and other text, which are marked as a "NOTE" or "EXAMPLE" (ECSS, 2009). This information is only for guidance in understanding, or for clarification of the associated requirement, therefore this text is not considered as a requirement. The expected output of requirement, such as "Real-time software dynamic design model [DDF, SDD; CDR]" is interpreted as the output is part of the DDF, contained in the SDD separated by comma and requested for the review CDR that is separated by semicolon. The sentences describing the requirement statements and associated information, such as identifiers, prerequisites/inputs and expected outputs are retrieved. The information, where necessary, is further followed in the normative and informative parts of the standard, for instance, to gather additional details of input and output work products, roles responsible for them, and applicability based on the certain tailoring criteria.

3.2 Structuring and Mapping Rules

For providing the convincing justification to the certification body about compliance means, the transformation of extracted information from standard documents into requirements, process models and compliance mappings is performed. In particular, the mapping rules are defined to structure and map the information (e.g., original requirements, inputs, outputs, tasks and roles) to the requirements and process models compliant with SPEM 2.0. Standards requirements and recommendations are mapped into requirements (i.e., *Practice* under the content packages) in SPEM 2.0. The statements or text leading to outputs or work products are not only mapped to the requirements, but also the tasks are created for them. If standard documents have clause, part-based distinction criteria and hierarchical levels, then the requirements are mapped as nested requirements (a requirement inside another requirement). Identifier with name, requirement description and objective are mapped into *Practice* fields name, brief description and purpose. Examples and note are mapped to the guidance kinds i.e., example and guideline that are linked to the corresponding task. The inputs or prerequisites for a specific requirement or task are mapped to the work products and linked to associated task as mandatory or optional inputs based on the keywords specified in the documents. Similarly, the expected outputs or deliverables are mapped to the work products. Typically, a task is assigned to a specific role, who is responsible for the execution of work, if document contained the role-related information then a role (e.g., supplier, requirements manager and designer) is created for the corresponding task. A new task is created for each piece of information or requirement. The hierarchical levels are also mapped to the process in SPEM 2.0. In particular, the main clause is mapped to the DeliveryProcess or the CapabilityPattern. The mapping

Algorithm 1: Generating the Requirements, Process Model and Compliance Mappings.

```
Input: Extracted Information of Standard Documents
Output: Requirements, Process Model, Compliance Mappings
while $ExtractedInformation \; != null$ do
    Transform
    $(ProcessComponent$ & $ContentPackage \leftarrow MainClause)$;
    for all Requirements do
        $(Practice$ & $Phase \mid Activity \leftarrow Structure)$;
        Link Practice to breakdownElements Activity or Phase $Practice.activityReferences$;
        for $Requirement$ in $ExtractedInformation()$ do
            $(subPractice$ & $Task \leftarrow Requirement)$;
            Link subPractice to Task $subPractice.contentReferences$;
            //Map <requirement>.puid to <element>.guid;
            <requirement>.id & <requirment>.name to <element>.name;
            <requirment>.description to <element>.briefDescription;
            <requirment>.objective to <element>.purpose;
        end for
        for $Input$ & $Prerequisite$ in $ExtractedInformation()$ do
            $(WorkProduct \leftarrow Input$ & $Prerequisite)$;
            //Link WorkProduct to Task as optional/mandatoryInput;
        end for
        for $Role$ in $ExtractedInformation()$ do
            $(Role \leftarrow Role)$;
            //Link Role to WorkProduct as responsibleFor;
        end for
        ...
    end for
end while
```

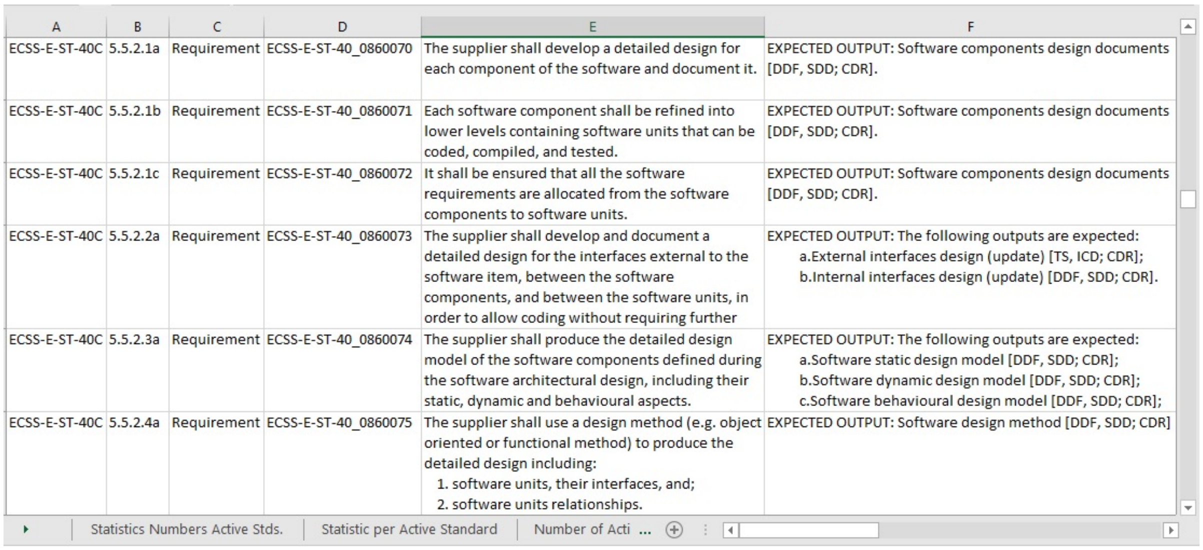
is focused on the Work Breakdown Structure of processes in EPF Composer. From there, the phases, activities and iterations are created such that structures are associated to phases, activities and so on. The activities are subsequently populated by applying tasks (as task descriptors) from the method content.

3.3 Generation of Requirements and Process Models

In this subsection, we present our algorithmic solution for the generation of the requirements and the process model compliant with SPEM in EPF Composer. This step is based on the structured information of standard documents. The mapping is focused on the method content elements and Work Breakdown Structure of process (decomposed linked elements, such as phases, activities etc.) in EPF Composer, as described in the previous subsection. Algorithm 1 shows the skeleton of our transformation. It starts by generating the ContentPackage and ProcessComponent for the main clauses. Then, it generates the content elements, such as tasks, roles, work products and guidance kinds practice (i.e., requirements) under ContentPackage. A delivery process in EPF Composer is contained in the metamodel class ProcessComponent, it is automatically created each time a delivery process or capability pattern is created. After that, all the phases, activities, iterations and milestones are created, and for each activity corresponding tasks are assigned. The corresponding process elements (i.e., phase, activity, task) are linked by using "activityReferences" and "contentReferences". The engineers can extend the process by providing additional information to the process description like its version, authors or team profiles required for the execution of the process. They may also dedicate their time for the manual production of portions of expected outputs/deliverables that require human intervention.

3.4 Reverse Compliance with Standard Documents

The reverse compliance of updated and extended process models is required during the execution phase to detect inconsistencies. To carry out the reverse compliance, the EPF composer models need to be processed in conjunction with the information extracted



A	B	C	D	E	F
ECSS-E-ST-40C	5.5.2.1a	Requirement	ECSS-E-ST-40_0860070	The supplier shall develop a detailed design for each component of the software and document it.	EXPECTED OUTPUT: Software components design documents [DDF, SDD; CDR].
ECSS-E-ST-40C	5.5.2.1b	Requirement	ECSS-E-ST-40_0860071	Each software component shall be refined into lower levels containing software units that can be coded, compiled, and tested.	EXPECTED OUTPUT: Software components design documents [DDF, SDD; CDR].
ECSS-E-ST-40C	5.5.2.1c	Requirement	ECSS-E-ST-40_0860072	It shall be ensured that all the software requirements are allocated from the software components to software units.	EXPECTED OUTPUT: Software components design documents [DDF, SDD; CDR].
ECSS-E-ST-40C	5.5.2.2a	Requirement	ECSS-E-ST-40_0860073	The supplier shall develop and document a detailed design for the interfaces external to the software item, between the software components, and between the software units, in order to allow coding without requiring further	EXPECTED OUTPUT: The following outputs are expected: a.External interfaces design (update) [TS, ICD; CDR]; b.Internal interfaces design (update) [DDF, SDD; CDR].
ECSS-E-ST-40C	5.5.2.3a	Requirement	ECSS-E-ST-40_0860074	The supplier shall produce the detailed design model of the software components defined during the software architectural design, including their static, dynamic and behavioural aspects.	EXPECTED OUTPUT: The following outputs are expected: a.Software static design model [DDF, SDD; CDR]; b.Software dynamic design model [DDF, SDD; CDR]; c.Software behavioural design model [DDF, SDD; CDR];
ECSS-E-ST-40C	5.5.2.4a	Requirement	ECSS-E-ST-40_0860075	The supplier shall use a design method (e.g. object oriented or functional method) to produce the detailed design including: 1. software units, their interfaces, and; 2. software units relationships.	EXPECTED OUTPUT: Software design method [DDF, SDD; CDR]

Statistics Numbers Active Stds.
Statistic per Active Standard
Number of Acti ...

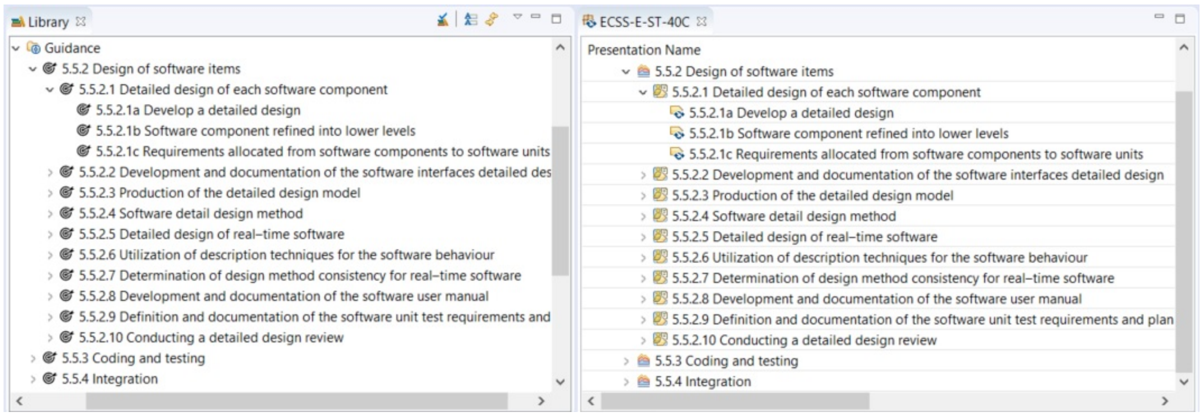
Figure 1: Portion of the ECSS EARM File.

from standard documents. This not just gives the possibility to verify the reverse compliance of extended models, but also the previously existing processes with the standard documents. It is noteworthy that the standards may not present all the information, such as used tools and roles. On the other hand, the customer or technical requirements and additional content elements are specified in process models. We focus on the delta analysis between models for identifying the gaps and measures for their resolution instead of the generation or replacement of extended or previously existing process models. In particular, the analysis is performed to identify the missing elements; for them, the potential candidates in process model are checked. The similarity assessment of missing and potential candidate elements is carried out based on their fields, such as name, description, roles and work products. The elements are added in an automatic manner after the approval from process engineer. Furthermore, the reverse compliance verification results include the list of elements containing sufficient and insufficient information (i.e., detected fallacies); appropriate recommendations to resolve the particular deviations are although provided, but they require manual effort for resolving them (Muram et al., 2018).

4 CASE STUDY

The ECSS Applicability Requirement Matrix (EARM) (ECSS, 2019) is Microsoft Excel file exported from the ECSS DOORS database containing all requirements with their identifiers, recommendations, expected outputs and permissions of the respective ECSS standard. The ECSS EARM file can be used for projects tailoring. Portion of the particular file is shown in Figure 1. ECSS-E-ST-40C standard (ECSS, 2009) is one of the series of European Cooperation for Space Standardization (ECSS) standards, which focuses on space software engineering processes requirements and their expected outputs. This standard contains one normative clause (clause 5), which defines the requirements for engineering software for space systems, applicable to any space projects producing computer software. In this paper, we limit our attention on clause 5.5 *software design and implementation engineering process* of ECSS-E-ST-40C standard, which consists of three phases (i.e., activity kinds): design of software items, coding and testing, and integration (ECSS, 2009), which may contain various activities. Each phase or activity in turn consists of one or more tasks, and each task is associated with one or more outputs, which are given in the *expected output* section.

The requirements related to the tasks are identified by a hierarchical number, made of the four digits of the clause (e.g., 5.5.2.1), followed by a letter (e.g., a, b or c) to further distinguish them. For instance, a requirement “The supplier shall develop a detailed design for each component of the software and document it” has an identifier “5.5.2.1a”. In circumstances when several outputs are expected from a task, they are identified by a letter (e.g., *a or b*) and the destination file of the output is indicated in brackets. The expected output of “5.5.2.1a” requirement is “software components design documents [DDF, SDD; CDR]”, as shown in Figure 1. This means that the output is the part of DDF (Design Definition File), contained


Library

- Guidance
  - 5.5.2 Design of software items
    - 5.5.2.1 Detailed design of each software component
      - 5.5.2.1a Develop a detailed design
      - 5.5.2.1b Software component refined into lower levels
      - 5.5.2.1c Requirements allocated from software components to software units
    - 5.5.2.2 Development and documentation of the software interfaces detailed des
    - 5.5.2.3 Production of the detailed design model
    - 5.5.2.4 Software detail design method
    - 5.5.2.5 Detailed design of real–time software
    - 5.5.2.6 Utilization of description techniques for the software behaviour
    - 5.5.2.7 Determination of design method consistency for real–time software
    - 5.5.2.8 Development and documentation of the software user manual
    - 5.5.2.9 Definition and documentation of the software unit test requirements and
    - 5.5.2.10 Conducting a detailed design review
  - 5.5.3 Coding and testing
  - 5.5.4 Integration

ECSS-E-ST-40C

Presentation Name

- 5.5.2 Design of software items
  - 5.5.2.1 Detailed design of each software component
    - 5.5.2.1a Develop a detailed design
    - 5.5.2.1b Software component refined into lower levels
    - 5.5.2.1c Requirements allocated from software components to software units
  - 5.5.2.2 Development and documentation of the software interfaces detailed design
  - 5.5.2.3 Production of the detailed design model
  - 5.5.2.4 Software detail design method
  - 5.5.2.5 Detailed design of real–time software
  - 5.5.2.6 Utilization of description techniques for the software behaviour
  - 5.5.2.7 Determination of design method consistency for real–time software
  - 5.5.2.8 Development and documentation of the software user manual
  - 5.5.2.9 Definition and documentation of the software unit test requirements and plan
  - 5.5.2.10 Conducting a detailed design review
- 5.5.3 Coding and testing
- 5.5.4 Integration

Figure 2: Generated Requirements and Process.

in the SDD (Software Design Document) separated by comma and requested for the review CDR (Critical Design Review) that is separated by semicolon. The work product details are further checked in the normative and informative parts of the standard. This is particularly relevant for detecting inconsistencies (or fallacies) during the reverse compliance checking.

The EARM file is processed to extract the information, in particular, requirement identifier (5.5.2.1a), requirement description and expected outputs are retrieved. The text is located in the ECSS-E-ST-40C standard document to obtain additional details. The section names are extracted for giving titles to the process component, phases and activities. For instance, “Design of software items” and “Detailed design of each software component” are extracted from the standard document. The requirements and process model are automatically generated in EPF Composer. In particular, four plugins are generated: (1) Customized_Icon plugin; (2) Standard requirements (ECSS-E-ST-40C_Requirements); (3) ECSS-E-ST-40C_Lifecycle plugin and (4) Mapping_Requirements plugin. The subsubsections in clause 5 “Requirements” are generated as high-level requirements i.e., Practices in EPF Composer (e.g., 5.5.2 Design of software items, 5.5.3 Coding and testing), whereas the next hierarchy level under these subsubsections is regarded as subrequirements (e.g., 5.5.2.1 Detailed design of each software component), as shown in Figure 2. Finally, the requirements that have outputs are created as subsubrequirements; their identifier has four digits and an alphabet.

Figure 2 shows the generated requirements and the process model compliant to the SPEM 2.0. The CapabilityPattern is created for the standard clause, in particular, a Work Breakdown Structure of process in EPF Composer. The high-level requirements are associated to phases, all subrequirements are associated to activities and subsubrequirements are associated to tasks. The expected outputs and responsible roles are linked to the corresponding task. The content elements such as tasks, work products, roles, which are the part of process are linked to the corresponding standard requirements through contentReferences; whereas the phases and activities in process are linked to the requirements through activityReferences. The generated requirements and process model can evolve during the life cycle. In the absence of support for automatic extraction of information from standard documents and generation of requirements, process models and compliance mappings, engineers would have to model them manually which is regarded as complex and labour-intensive. During the reverse compliance management, the missing process elements and insufficient information (i.e., fallacies) are detected in process model and corresponding measures for their resolution are informed to the engineers. The missing elements are automatically added after approval; however, the insufficient information need to be manually added by engineers.

5 RELATED WORK

The discussion of related work concerns two topics: process extraction and compliance management.

5.1 Process Extraction

The work presented in (Friedrich et al., 2011) utilised the existing syntax parsing and semantic analysing mechanisms in combination with anaphora resolution for generating a BPMN model. In particular, a model that includes the extracted actions, actors and flows is generated. However, the processing of paragraphs and tables were not considered. Leopold et al. (Leopold et al., 2014) presented an approach that transforms the BPMN models into natural language texts. The proposed approach is based on existing parsing and annotation technique, and refined process structure tree. van der Aa et al. (van der Aa et al., 2019) focused on the automated extraction of declarative process models from textual descriptions. The authors extended existing natural language processing techniques in order to identify activities and their inter-relations in constraint descriptions and transform these into declarative constraints. Yanuarifiani et al. (Yanuarifiani et al., 2019) proposed a methodology for mapping the requirements ontology to BPMN elements and generating the BPMN XML file. The requirements ontology consists of three classes: action, position and object. Qian et al. (Qian et al., 2020) proposed a hierarchical neural network to extract process models from process texts. They focused on the coarse-to-fine learning mechanism, training multi-grained tasks in coarse-to-fine grained order, to apply the high-level knowledge for the low-level tasks. Previous studies have just focused on the extraction and generation of BPMN model elements. In this paper, we targeted the automatic extraction of SPEM-based requirements and a broad set of process elements, such as tasks, roles, work products, guidance and iteration. They are generated in EPF Composer.

5.2 Compliance Management

Sànchez-Ferreres et al. (Sànchez-Ferreres et al., 2017) used existing natural language processing techniques for aligning a textual description and a process model. They converted both inputs into an identical representation of feature vectors and compared by means of standard distance metrics. Winter et al. (Winter et al., 2020) also exploited the natural language processing to enable assessment of compliance between a set of paragraphs from a regulatory document and BPMN models. Delicado et al. (Delicado et al., 2017) introduced an online platform called NLP4BPM that converts a textual description into a BPMN model and vice versa. The platform also computes the alignment between two BPMN models and a textual description with a BPMN model. The presented interface adapted or modified techniques appeared in the last years (Friedrich et al., 2011; Leopold et al., 2014; Sànchez-Ferreres et al., 2017).

Ardila et al. (Ardila et al., 2018) presented the transformation of SPEM 2.0-compatible requirements and process models into input models of Regorous to perform compliance checking. However, not only standards information is manually extracted but also requirements and process models are manually modelled in EPF Composer. Jiang et al. (Jiang et al., 2015) proposed a consistency and compliance checker framework that analyses the consistency of a set of regulations itself and also verifies the business process compliance with a set of interrelated regulations. However, the translation of natural language regulations into a formal model is manually performed. This paper targets the automated processing of standard documents, generation of requirements, process models and compliance mappings at planning phase, as well as reverse compliance management during execution phase.

6 CONCLUSION AND FUTURE WORK

To support the process compliance, which is mandatory for several industrial domains, the central theme of this paper focuses on four particular aspects: (i) natural language processing of standard documents for automated extraction; (ii) structuring of requirements and process models by considering the SPEM based mapping rules; (iii) generation of requirements, process models and compliance mappings in EPF Composer; and (iv) reverse compliance management of process models. The natural language processing of standard documents is carried out for extracting the requirements and content elements of processes, such as tasks, roles and work products. The requirements are organized into a hierarchy, while the content elements of process are organized into a work breakdown structure by considering the phases, activities and milestones. During the planning phase, the requirements, process models and compliance mappings are generated in EPF Composer. In the context of reverse compliance that can be carried out during execution phase, the gaps with extended or pre-exiting process models are detected and resolution measures are provided; missing elements are added based on approvals. The application of the proposed methodology is illustrated for the ECSS-E-ST-40C compliant space system engineering.

This research is primarily based on the natural language processing investigations that are performed for the ECSS-E-ST-40C standard. However, there is a need to conduct further evaluations for the process extraction and reverse compliance management with other standards. In the future, we intend to support the extraction of process models from digital twins (Muram et al., 2020) and their reverse compliance management. The processing of assurance (safety and security) cases in free text and tabular representations, and generation of Goal Structuring Notation (GSN) is also a part of our future work agenda. Another direction for future work is to achieve the synchronisation between process, product and assurance cases.

ACKNOWLEDGEMENTS

This work is partially supported by FiC (Future factories in the Cloud) project funded by SSF (Swedish Foundation for Strategic Research). The second author has also participated during the tenure of an ERCIM "Alain Bensoussan" Fellowship Programme.

REFERENCES

Ardila, J. P. C., Gallina, B., and Muram, F. U. (2018). Transforming SPEM 2.0-compatible process models into models checkable for compliance. In *International Conference on Software Process Improvement and Capability Determination, SPICE '18, Tessaloniki, Greece*, pages 233–247.

Delicado, L., Sànchez-Ferreres, J., Carmona, J., and Padró, L. (2017). NLP4BPM - natural language processing tools for business process management. In *15th International Conference on Business Process Management, BPM '17 Demo Track, Barcelona, Spain*.

ECSS (2009). European Cooperation for Space Standardization, ECSS-E-ST-40C, Space Engineering Software.

ECSS (2019). European Cooperation for Space Standardization, ECSS Applicability Requirement Matrix (EARM). https://ecss.nl/standards/downloads/earm/.

Friedrich, F., Mendling, J., and Puhlmann, F. (2011). Process model generation from natural language text. In *23rd International Conference on Advanced Information Systems Engineering, CAiSE '11, London, UK*, pages 482–496.

Javed, M. A. and Gallina, B. (2018). Safety-oriented process line engineering via seamless integration between EPF composer and BVR tool. In *22nd International Systems and Software Product Line Conference - Volume 2, SPLC '18, Gothenburg, Sweden*, pages 23–28.

Javed, M. A., Gallina, B., and Carlsson, A. (2019). Towards variant management and change impact analysis in safety-oriented process-product lines. In *34th ACM/SIGAPP Symposium on Applied Computing, SAC 2019, Limassol, Cyprus*, pages 2372–2375.

Javed, M. A., Muram, F. U., and Zdun, U. (2018). On-demand automated traceability maintenance and evolution. In *17th International Conference on New Opportunities for Software Reuse, ICSR '18, Madrid, Spain*, volume 10826, pages 111–120.

Javed, M. A., Stevanetic, S., and Zdun, U. (2016). Towards a pattern language for construction and maintenance of software architecture traceability links. In *21st European Conference on Pattern Languages of Programs, EuroPLoP '16, Kaufbeuren, Germany*, page 24.

Jiang, J., Aldewereld, H., Dignum, V., Wang, S., and Baida, Z. (2015). Regulatory compliance of business processes. *AI Soc.*, 30(3):393–402.

Leopold, H., Mendling, J., and Polyvyanyy, A. (2014). Supporting process model validation through natural language generation. *IEEE Trans. Software Eng.*, 40(8):818–840.

Muram, F. U., Gallina, B., and Kanwal, S. (2019). A tool-supported model-based method for facilitating the EN50129-compliant safety approval process. In *Third International Conference on Reliability, Safety, and Security of Railway Systems. Modelling, Analysis, Verification, and Certification, RSSRail '19, Lille, France*, pages 125–141.

Muram, F. U., Gallina, B., and Rodriguez, L. G. (2018). Preventing omission of key evidence fallacy in process-based argumentations. In *11th International Conference on the Quality of Information and Communications Technology, QUATIC '18, Coimbra, Portugal*, pages 65–73.

Muram, F. U., Javed, M. A., Hansson, H., and Punnekkat, S. (2020). Dynamic reconfiguration of safety-critical production systems. In *25th IEEE Pacific Rim International Symposium on Dependable Computing, PRDC '20, Perth, Australia*, pages 120–129.

OMG (2008). Object Management Group, Software & Systems Process Engineering Metamodel Specification (SPEM), Version 2.0. http://www.omg.org/spec/SPEM/2.0/.

OMG (2011). Object Management Group, Business Process Model And Notation (BPMN), Version 2.0. https://www.omg.org/spec/BPMN/2.0.

Qian, C., Wen, L., Kumar, A., Lin, L., Lin, L., Zong, Z., Li, S., and Wang, J. (2020). An approach for process model extraction by multi-grained text classification. In *32nd International Conference on Advanced Information Systems Engineering, CAiSE '20, Grenoble, France*, pages 268–282.

Sànchez-Ferreres, J., Carmona, J., and Padró, L. (2017). Aligning textual and graphical descriptions of processes through ILP techniques. In *29th International Conference on Advanced Information Systems Engineering, CAiSE '17, Essen, Germany*, pages 413–427.

van der Aa, H., Ciccio, C. D., Leopold, H., and Reijers, H. A. (2019). Extracting declarative process models from natural language. In *31st International Conference on Advanced Information Systems Engineering, CAiSE '19, Rome, Italy*, pages 365–382.

Winter, K., van der Aa, H., Rinderle-Ma, S., and Weidlich, M. (2020). Assessing the compliance of business process models with regulatory documents. In *39th International Conference on Conceptual Modeling, ER '20, Vienna, Austria*, volume 12400, pages 189–203.

Yanuarifiani, A. P., Chua, F., and Chan, G. (2019). Automating business process model generation from ontology-based requirements. In *8th International Conference on Software and Computer Applications, ICSCA '19, Penang, Malaysia, February 19-21*, pages 205–209.