Ensuring Conformance to Process Standards Through Formal Verification

Edward Kabaale[2(✉)], Lian Wen[1,2], Zhe Wang[1,2], and Terry Rout[1]

[1] Institute for Integrated and Intelligent Systems, Griffith University, 170 Kessels Road, Nathan, QLD 4111, Australia
{l.wen,z.wang,t.rout}@griffith.edu.au

[2] School of Information and Communication Technology, Griffith University, 170 Kessels Road, Nathan, QLD 4111, Australia
edward.kabaale@griffithuni.edu.au

Abstract. Software process standards and models encapsulate best practices and guidelines for engineering and managing software. These are usually prescribed in natural language. However, natural language based process specifications can be inconsistent and ambiguous that makes it difficult to monitor and verify if they have been fully implemented and adhered too in a given software project. Besides the process of defining and documenting the necessary evidence to comply with process standard requirements is often manual, time consuming and laborious. In earlier studies, we developed a translation scheme and metamodel for consistent and uniform software process formalisation. In the current study, we leverage the formal process specification to develop a two-step formal process verification approach; first we extract process requirements from the standard documents and translate them into logical axioms. We then augment these axioms with additional information in a process verification ontology. This ontology is then utilised in conformance verification of a performed process. We demonstrate the feasibility of our approach with software requirements analysis process and a case study.

Keywords: Process · Standards · Ontology · Verification

1 Introduction

Software process management is a systematic and continuous endeavor to define, assess and improve processes that are used to produce quality software products and services within the constraints of time, budget and schedule [1]. The process perspective to software development is premised on the manufacturing principle that product quality is influenced and evolved by the process used to produce it [1]. To systematise software development and ensure interoperability, consistency, and repeatability; software process standards such as ISO/IEC 12207 [2], ISO/IEC 29110 [3] are widely adopted as a source of universally accepted best

I. Stamelos et al. (Eds.): SPICE 2018, CCIS 918, pp. 248–262, 2018.
https://doi.org/10.1007/978-3-030-00623-5_17

practices and guidelines to support the design, implementation and improvement of software processes. However, software process standards are usually prescribed in natural language that makes it difficult to implement, monitor and verify such processes in practice [4]. Given that software standards are typically multi-paper documents and vebrose, they are likely to be ambiguous and inconsistent which impedes their automated analysis and verification. Novice users may find them hard to implement and verify in projects [5]. Despite, some efforts making software standards more applicable and accessible even to very small entities (VSE); for example ISO/IEC 29110 with deployment packages[1], still considerable time and resources are needed to understand, implement and verify thier conformance [6].

The process of verifying the extent to which the implemented process is in conformance with a process reference model (PRM) is referred to as process compliance [7] and accomplished through *process assessment* or appraisal [8]. Process assessment is the disciplined evaluation of the organisational processes against a set of criteria defined in a process assessment model (PAM) to determine the capability of the organisational processes to perform within the constraints of quality, cost and schedule [8]. During process assessment, the emphasis is placed on two measures, i.e., organisational process conformance to the PRM and the effectiveness of the organisational processes (process capability) to achieve organisational business objectives [8]. Where as a process capability assessment studies individual processes and thier attributes, a conformance assessment on the other hand, can study fulfillment of a standard's requirements [9,10]. Therefore, Process assessment provides a way to verify conformance to a standard like ISO/IEC 29110, if such a standard is considered a set of requirements [10]. The main stream assessment methods such as ISO/IEC 15504 (aka SPICE) that is transiting to ISO/IEC 330xx [11] describe guidelines that standards compliant processes should follow namely; (i) defining processes in terms of purpose and outcomes, (ii) use of objective evidence to prove conformance to the defined criteria. When assessing software process implementations, the main stream assessment methods are however, complex, resource intensive and unaffordable to many software companies [12].

To overcome the above challenges, formal approaches to process modeling and verification have been proposed in previous studies [4,7,12–14]. These increase confidence and trustworthiness in the evidence used for process compliance [15] and enable the use of automated analysis and verification techniques in software process [17]. A formal process specification also enables compliance and certification to process standards and reference models [18]. However, we couldn't use the available approaches for the task at hand for various reasons; The approach by [7] employs first order logic (FOL), to formalize and verify standards compliant software development. FOL is a proven and necessary expressive formalism. However, its major drawback is its undecidability that leads to inefficiency in terms of computational costs. Other approaches are not expressive enough for the task at hand.

[1] http://profs.etsmtl.ca/claporte/english/vse/vse-packages.html.

Essentially ontology approaches are an application of formal methods into the semantic web where web resources are formally specified using logical notations and rigorously verified using ontology reasoning engines [14]. In recent years, Description Logics (DLs) [16] based ontologies have been widely accepted as an important means for representing and formalising knowledge in different domains including software process engineering (see, e.g., [17,19]). DLs are a decidable fragment of FOL that provides a rich and flexible modeling language that underpins the web ontology language (OWL)[2]; a W3C standard for developing ontologies in the semantic web. DLs come with an unambiguous, standardised semantics and a wide range of tools that can be used to develop, validate, integrate and verify formal models of software processes. Moreover, DLs are supported by a variety of optimised inference engines[3] that can be utilised to support both consistent process implementation and querying the process space by logical expressions, e.g. in conformance checking.

In earlier studies [20,21], we developed a translation scheme and metamodel for consistent and uniform process formalisation. In the current study, we leverage such a formal process specification for process verification since formal models are basically a set of domain theorems that are amenable to formal proving through reasoning [14]. Therefore in this paper, we extend our earlier work, with a formal process verification approach where two levels of verification can be performed to ensure the correctness of a process specification, i.e., the ontology and instance verification levels. The former ensures the correctness of the process specification itself, and the latter ensures the conformance of a process instance to the standard process. Moreover, ontology reasoners can be used to fully automate these formal verification activities. Consequently, we develop a process verification ontology where we treat the problem of conformance to standard processes as *instantiation* with a case study. The paper is structured as follows: In Sect. 2 we provide preliminaries about process management and improvement, a formal foundation about description logic and ontologies while we present an overview of our approach in Sect. 3 and a feasibility study of this approach based on software requirements analysis process. Simultaneously, we provide challenges for further development and evaluation of the approach in Sect. 4 concludes the paper.

2 Background

2.1 Process Management and Improvement

Software process management is the use of process engineering concepts, techniques, and practices to explicitly monitor, control, and improve the software process [1]. The objective of software process management is to enable an organization to produce software products according to a plan while simultaneously improving the quality of its products [1]. A process is a *set of interrelated or*

[2] https://www.w3.org/OWL/.
[3] http://owl.cs.manchester.ac.uk/tools/list-of-reasoners/.

interacting activities which transforms inputs into outputs [2]. We limit ourselves to defining a process in terms of its purpose and outcomes [8]. Processes be technical or management are an inherent part of software engineering (SE), so is process assessment which is a foundation step for process improvement [29]. Organisations use process assessment models (PAM) such as ISO/IEC 330xx [11] to evaluate and change their processes in light of achieving business objectives and supporting process conformance to standard processes [10]. A PAM is a two dimensional representation that describes processes in terms of objective evidence that may be identified to demonstrate process implementation. PAMs generally comprise sets of practices and descriptions of work products that serve as indicators of process performance and process capability dimensions of the assessment framework [2]. The process dimension of a PAM describes processes drawn from one or more PRMs, in SE, these processes can be drawn from ISO/IEC 12207, ISO/IEC 29110 among others. The PRM provides a list of processes to be verified and their descriptions with a common terminology and scope for process assessments. In a PRM, processes are described in terms of purpose and outcomes. According to [9,10] conformance assessment may utilize PRM in evaluating achievement of the process outcomes. We use software requirements analysis (SRA) process from ISO/IEC 15504-5 [22] as a running example in this paper.

Process Purpose: *The purpose of the Software requirements analysis process is to establish the requirements of the software elements of the system.*

Process Outcomes:

- *PO1: The requirements allocated to the software elements of the system and their interfaces are defined;*
- *PO2: Software requirements are analysed for correctness and testability*
- *PO3: The impact of software requirements on the operating environment are understood*
- *PO4: Consistency and traceability are established between the software requirements and system requirements*
- *PO5: Prioritization for implementing the software requirements is defined*
- *PO6: The software requirements are approved and updated as needed*
- *PO7: Changes to the software requirements are evaluated for cost, schedule and technical impact*
- *PO8: The software requirements are baselined and communicated to all affected parties*

Accordingly, SRA process outcomes can be achieved by implementing *base practices* and evidencing their implementation through availability of *work products* produced. These are also referred to as the process performance assessment indicators for the process dimension in the process assessment model. The base practices and work products for software requirements analysis process drawn from [22] are shown in Tables 1 and 2 respectively.

ISO/IEC 33020 [23] defines an ordinal scale for the evaluation of process capability based upon six defined capability levels. It characterises the extent to which

Table 1. SRA process base practices

No	Base practices	Process outcome achieved
BP1	Specify software requirements	PO1, PO2, PO5
BP2	Determine operating environment impact	PO3
BP3	Develop criteria for software testing	PO2
BP4	Ensure consistency	PO4
BP5	Evaluate and update software requirements	PO6, PO7
BP6	Communicate Software requirements	PO8

Table 2. SRA Process process work products

No	Work product	Process outcome evidenced
WP1	Communication record	PO8
WP2	Change control	PO7
WP3	Traceability record	PO4
WP4	Analysis report	PO2, PO3, PO7
WP5	Interface requirements	PO1
WP6	Software requirements	PO1, PO2, PO4, PO5 and PO6

the process outcomes are achieved. The outcome achievement is behaviourally aggregated to the process attribute which in turn is transformed to an ordinal scale as shown in Table 3 and aggregated to determine a given capability level [24]. We use this rating scheme in a *conjugate* way [24] to determine the extent to which the process outcomes are achieved.

2.2 Description Logics and Ontologies

Ontologies are engineering artefacts that are an explicit formal specification of a shared conceptualisation [25]. Web Ontology Language (OWL)[4] is an ontology modeling language designed and standardised by W3C for modeling ontologies in the semantic web. OWL is underpinned by DL for its formal semantics [16] that enables expressed knowledge to be reasoned on by human and artificial agents. An OWL DL ontology is mainly composed of two main components; The Terminological knowledge represented in the TBox (Class Level) and the Assertional knowledge forming the ABox (Instance Level). The TBox defines the intensional knowledge by which a concrete world can be described in form of classes, properties and the respective axioms that define the constraints on the conceptual schema. The ABox on the other hand, represents assertional knowledge that describes some particular situation that instantiates the TBox. Although in DL, only the Tbox is commonly referred to as an ontology and the

[4] https://www.w3.org/OWL/.

Table 3. The rating scale of process outcomes (cited from ISO/IEC 33020 [23])

Acronym	Achievement of the defined Outcome
Not achieved (**N**)	There is little or no evidence of achievement of the defined outcome in the assessed process
Partially achieved (**P**)	There is some evidence of the approach to, and some achievement of, the defined attribute in the assessed process
Largely achieved (**L**)	There is evidence of a systematic approach to, and significant achievement of, the defined outcomes in the assessed process. Some weakness related to this attribute may exist in the assessed process
Fully achieved (**F**)	There is evidence of a complete and systematic approach to, and full achievement of, the defined outcome in the assessed process. No significant weaknesses related to the outcome exist in the assessed process

combination of the TBox and ABox is referred to as the knowledge base (KB), in this work we use both ontology and knowledge base interchangeably. In the scope of our solution, the actual process implementation will be treated as the ABox while the process standard will be coded as the TBox.

Ontology modeling in the semantic web, follows an open world assumption (OWA) where anything is permissible unless explicitly prohibited. In others, OWA semantics assume incomplete information by default, i.e., missing information is treated as unknown rather than false. If any information is not declared in the ontology, it is not taken to be false as is the case in database systems that follow a closed world assumption (CWA) [26]. DL Axioms are used to constrain classes, properties and individuals that classes can admit. Where as axioms in an OWA semantics are used for inference purposes, in this study, we would like such axioms to behave more like integrity constraints [26] in the presence of process instance data (ABox) that needs to conform to the standard process in the TBox (ontology) through the verification process using an ontology reasoner such as Pellet [27]. Indeed, whereas inference is useful for reasoning over the domain knowledge, when dealing with process conformance, the assessor wants to verify that the presented objective evidence in form of e.g., artefact such as SRS is indeed validated and baselined.

3 Overview of the Approach

In this section, we present our approach that includes translation of natural language process to formal specification (DL language), capturing process standard requirements as DL axioms consisting of base practices, work products and process outcomes. We also present a process instance in a form of an ABox that we verify against the DL Axioms in the TBox using a reasoner such as Pellet.

The consistency of which represents conformance to the standards requirements or non conformance. Figure 1, gives a high level view of the process verification approach presented in this section.

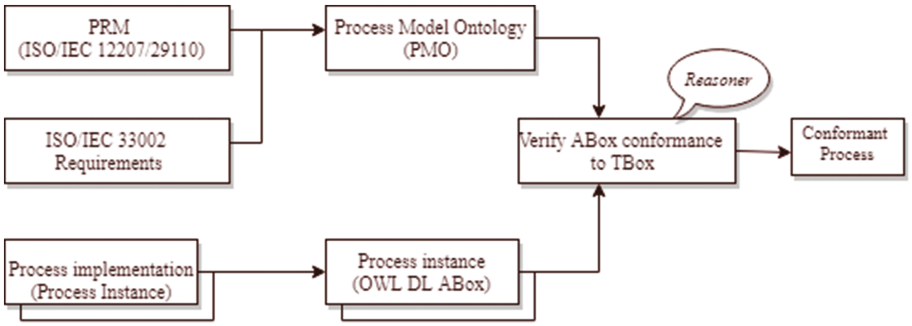

Fig. 1. Process verification architecture

3.1 Constructing the Process Model Ontology (PMO)

In [29] a conceptual framework for ontology usage in process assessment is provided. In this framework a number of ontologies are proposed such as PRM ontology, measurement and process ontology. In the current study, we mainly develop a process model ontology (TBox)consisting the PRM and PAM, and the process ontology as the process instance (ABox) which we use for process verification. A well known ontology construction approach recommends investigating available upper ontologies and extracting reusable concepts from them before developing domain specific ontologies. We refer to the upper ontology developed for ISO software engineering standards [28] that provides an ontological base for all ISO present and future standards. This ontological infrastructure provides an ontological base for various standard domain ontologies that are specific to a given domain such as process management. We also make use of concepts and properties in earlier developed ontologies for the SE domain such as the Software Lifecycle Ontology (SLO) and Software Implementation Process Ontology (SIP) developed in the ALIGNED project[5] that we reuse in our ontology development following guidelines suggested in [29]. The ontology we develop is in conformance with the guidelines prescribed for process descriptions in ISO/IEC TR 24774 [30]. The PMO ontology is defined via the concepts and roles that describe the process in terms of objective evidence that prove process performance in an organisation.

A key concept of the ontology is Process that is defined through its intended purpose and evidenced by the achievement of the process outcomes. Process

[5] http://aligned-project.eu.

outcomes are sufficient and necessary conditions to achieve the software purpose. To achieve the ProcessOutcomes, Practices (i.e., base and generic) are performed to produce WorkProducts. WorkProducts are also inputs to the Practices that are used to achieve the ProcessOutcomes. WorkProducts are used to prove that the practices are being implemented and ProcessOutcomes are being achieved. For instance, in our running example, the SRS a type of work product can be used to prove that the software requirements analysis process was carried out. The ontology is further augmented with DL axioms to constrain the class behaviours and the instances that can be admitted by the ontology.

3.2 Representing Standard Requirements as DL Axioms

Generally standards can be considered as a set of requirements (rules) prescribing what should be done in order to achieve the process outcomes [31]. ISO/IEC 33002 constrains processes in a PRM to be defined in terms of purpose and outcomes. Where a set of process outcomes is necessary and sufficient to achieve the process purpose. In order to achieve the process outcomes, ISO/IEC 15504-5 [22] specifies base practices to be implemented with work products providing evidence that the base practices are being performed and the outcomes are being achieved. Where as conformance assessment may utilize the PRM in evaluating the achievement of process outcomes [10], there is no guarantee that individual process outcomes are being achieved. To overcome this situation, we extract the process requirements from the standard documents in form of $if...then$ statements made up of three major components; practices, work products and process outcomes that are translated into DL axioms. These constitute statements such as; *If a given base practice is implemented and evidenced by a work product then a related process outcome should be achieved.* These are illustrated with a set of SRA process outcomes, practices and work products identified in Sect. 2 we slightly adapt the outcome numbering from ISO/IEC 15504-5 to PO1..., for uniformity through out the example illustrations in this paper.

1. If `SRA` has software requirements that are specified in a `SRS`, then outcome `PO1` is achieved

$$SRA \sqcap \forall hasSR.(\exists specifiedIn.SRS) \sqsubseteq achieve.\{PO1\} \tag{1}$$

2. If `SRA` has `TestingCriteria` for software requirements that is developed and recorded in an `AnalysisReport`, then outcome `PO2` is achieved

$$SRA \sqcap \forall hasSR.((\forall hasTestingCriteria.Developed) \sqcap \exists recordedIn.AnalysisReport) \sqsubseteq achieve.\{PO2\} \tag{2}$$

3. If `SRA` has the impact of software requirements on the operating environment that is determined and recorded in analysis report, then outcome `PO3` is achieved

$$SRA \sqcap \forall hasSR.((\forall hasImpact.Determined) \sqcap \exists recordedIn.AnalysisReport) \sqsubseteq achieve.\{PO3\} \tag{3}$$

4. If SRA has consistency of system requirements (SSR) to software requirements that is ensured and recorded in a traceability record, then outcome PO4 is achieved

$$SRA \sqcap \forall hasSR.((\exists consistencyEnsuredBetween.SSR) \sqcap \exists recordedIn.AnalysisReport) \sqsubseteq achieve.\{PO4\} \quad (4)$$

5. If SRA has software requirements that are prioritised and documented in SRS, then outcome PO5 is achieved

$$SRA \sqcap \forall hasSR.(\exists prioritisedIn.SRS) \sqsubseteq achieve.\{PO5\} \quad (5)$$

6. If SRA has software requirements that are evaluated and approved by the customer and updated in SRS, then outcome PO6 is achieved

$$SRA \sqcap \forall hasSR.(\exists evaluatedWith.Customer \sqcap \exists updatedIn.SRS) \sqcap \exists recordedIn.AnalysisReport \sqsubseteq achieve.\{PO6\} \quad (6)$$

7. If SRA has changes to the requirements that are evaluated for cost, schedule and technical impact and recorded in change control record and analysis report, then outcome PO7 is achieved

$$SRA \sqcap \forall hasSR.(\forall hasChange.Evaluated \sqcap \exists recordedIn. (ChangeControl \sqcap AnalysisReport) \sqsubseteq achieve.\{PO7\} \quad (7)$$

8. If SRA has software requirements that are communicated to all parties who will use them and recorded in a communication record, then outcome PO8 is achieved.

$$SRA \sqcap \forall hasSR.(\exists communicated \sqcap recordedIn.CommunicationRecord) \sqsubseteq achieve.\{PO8\} \quad (8)$$

To determine the extent to which the process outcomes are achieved, we follow the process rating scheme from ISO/IEC 33020 as shown in Table 3. Since SRA process has eight process outcomes, we base our rating scheme on the achievement of these outcomes as shown in DL axioms (9)−(13). Given a SRA process if two but less than four Process Outcomes are achieved this will be ranked as the process not being achieved.

$$NotAchieved \equiv SRA \sqcap (\geqslant 2 \sqcap < 4) achieve.ProcessOutcomes \quad (9)$$

Given a SRA process if four but less than six process outcomes are achieved, then the process will be ranked as partially achieving its purpose.

$$PartiallyAchieved \equiv SRA \sqcap (\geqslant 4 \sqcap < 6) achieve.ProcessOutcomes \quad (10)$$

Given a SRA process if six but less than eight process outcomes are achieved, then the process will be ranked as largely achieving its purpose.

$$LargelyAchieved \equiv SRA \sqcap (\geqslant 6 \sqcap < 8) achieve.ProcessOutcomes \quad (11)$$

Given a SRA process and it achieves all its eight process outcomes, then the process will be ranked as fully achieving its purpose.

$$FullyAchieved \equiv SRA \sqcap (\geqslant 8) achieve.ProcessOutcomes \tag{12}$$

Achieving process outcomes to the level of largely or fully achieved status, helps to build process performance attribute (PA1.1) that forms capability level one of the assessed process.

$$\begin{aligned} CapabilityLevelOne \equiv \exists hasProcessAttribute \cdot (PA1.1 \sqcap \\ hasRating \cdot (LargelyAchieved \sqcup FullyAchieved)) \end{aligned} \tag{13}$$

3.3 DL ABox Construction (Moodle SRA Process Instance)

As an implementation process instance, we adapt and give a detailed analysis and verification of the moodle e-learning system software requirements analysis process suggested in [32]. In this case study, we only highlight and summarise activities related to SRA process in Table 4, for other software implementation processes, interested readers can refer to [32]. In a moodle SRA process, a moodle community starts by communicating with the core team for performing a *feature voting activity*. The features could be about functional requirements (*req1*) or performance requirements (*req2*). In Moodle, there are four activities to be executed in order to vote and select a feature and develop a requirement specification for the selected feature(s). They are *feature voting activity*, *road map development*, *developing the requirements specifications* and *suggestions, discussions and agreements about the requirements specification*. At each end of a successful execution of these activities, a work product is produced. For example the *roadmap list* is developed after successful completion of voting process for selecting and prioritizing the features with the highest number of votes. The roadmap is used to guide the implementation of selected features. In Moodle, *software requirements specification (SRS) documents* are to be created for each of the feature added to the roadmap. The final work product in moodle SRA process are the suggestions and discussion on the *SRS* document that the entire community provides and agrees to, based on the specification released earlier. Due to space constraints, we only provide summarised process evidence for the first two TBox axioms (1–2) in form of DL ABox identified during Moodle development process in Table 6. The ABox for the remaining TBox axioms (3–8) can be constructed in the same way.

3.4 Process Verification

During software process verification, the object that is checked and verified is the process instance against the standard process using objective evidence from the organisation that indeed shows the process was carried out by the organisation. A process instance (PI) is defined to be a singular instantiation of a process that is uniquely identifiable and about which information can be gathered in a

Table 4. Moodle SRA process terminology

Moodle concept	Abox concepts
Moodle software requirements analysis process	*mSRA*
Software requirements	*SR*
Moodle software requirements specification	*mSRS*
Requirement examples	*req1* and *req2*
Roadmap	*rm*
System requirements	*ssr*
Moddle community	*mc*
Open source forums such as blogs, email	*openforum*

Table 5. Moodle SRA process evidence

Process outcome	Moodle SRA process evidence
PO1	Moodle roadmap created
PO2	Feature voting process
PO3	Feature voting process
PO4	-
PO5	Moodle SRS created
PO6	Moodle SRS is discussed and agreed upon
PO7	Moodle SRS discussions and agreement
PO8	Requirements are communicated through open forums and road map

Table 6. Moodle process instance (Abox)

Instance	Relations	Process outcomes
`SRA`(*mSRA*)	hasSR(*mSRA, req1*)	achieve(*mSRA, P01*)
`SR`(*req1*)	hasSR(*mSRA, reg2*)	
`SR`(*req2*)	specifiedIn(*req1, mSRS*)	
`SRS`(*mSRS*)	specifiedIn(*req2, mSRS*)	
SRA(mSRA)	hasSR(mSRA, req1)	achieve(mSRA, P02)
SR(req1)	hasSR(mSRA, req2)	
SR(req2)	hasTestingCriteria(req1, tc1)	
Developed(tc1)	hasTestingCriteria(req2, tc2)	
Developed(tc2)	recordedIn(tc1,rm	
TestingCriteria(tc1)	recordedIn(tc2, rm)	
TestingCriteria(tc2)		
AnalysisReport(rm)		

repeatable manner [22]. In this case study, we take moodle SRA process as a process instance about which information has been gathered and summarised in a form of an ontology ABox in Table 6. From this example, we can use an ontology reasoner such as pellet to verify automatically if moodle process (ABox) is conformant to the SRA process represented in the TBox axioms (1)–(8). If the moodle process achieves the rating of largely or Fully achieved, then it can be ranked as a performed process (see axiom (13)). Process performance is the process attribute that helps to build capability level one of the assessed process [2]. Therefore moodle process achieves its purpose and is classified at capability level one see Fig. 2. An ABox inconsistency with the TBox axioms can vary the process ratings (DL Axioms 9–12) based on the process outcomes achieved. Based on the analysis results in Table 5, there was no moodle process evidence to support the achievement of SRA process outcome PO4, i.e., *consistency and traceability are established between software requirements and system requirements*. Given our rating scheme for process outcome achievement in axioms (9)−(12), moodle process SRA process is rated as largely achieved.

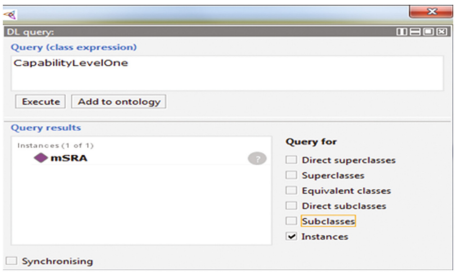

Fig. 2. Inferred mSRA process as an instance of CapabilityLevelOne

In order to test our process verification approach and the inferencing services offered by ontologies, we have implemented the ontology in Protege[6], a free open source editor for building and storing ontologies based on OWL standard. Protege integrates different reasoners such as pellet as well as different visualization schemes that we have used in our approach. DL facilitates different reasoning services both at the ontology and instance level. In our software process verification example, we used them both for consistency and process conformance checking via the DL query tab in protege see Fig. 2.

4 Conclusion and Future Works

In this paper, we proposed a formal approach to software process analysis and verification using Description logic based ontology. This paper is a significant

[6] https://protege.stanford.edu/.

extension to our previous works [20,21] by providing a more systematic, consistent and practical approach to translate software processes into DL ontology so that they can be formally reasoned and verified by using ontology reasoners, thereby automating the conformance approach through formal verification. We have represented software process standard requirements as DL axioms argumenting them with other process information in a PRM ontology (DL TBox). We also represented the moodle e-leaning system case study as a DL ABox specifying the evidence derived during moodle software development. Using ontology reasoners we are able to automatically perform confirmatory verification of process implementation (moodle case study) and establish that moodle case study is classified at capability level one. This means that the moodle process is able to achieve its process purpose through process outcome achievement. Ontology reasoners enable the process to be automated improving the efficiency, effectiveness and reduce on human errors in process assessments and improvement.

Future work includes extending the approach to the capability dimension of the PAM. We also intend to further evaluate the developed ontology using competency questions. These can be drawn from process assessment questionnaires and translated to SPARQL[7] queries to derive more information from the ontology. Automating the process parsing software process requirements from textual descriptions (natural language standards) into DL axioms is another area for future work. This is done manually at the moment and without a predefined approach. We could systematise it through the use of natural language process techniques. Secondly, we plan to develop a graphical user interface (GUI) to support users during process analysis and verification. This tool will be based on the features highlighted in [29]. The tool should integrate all the formal model definitions and verification steps proposed in this paper. We also intend to carry out more case studies in industry.

References

1. Kitson, D., Humphrey, W.S.: The role of assessment in software process improvement. Technical report CMU/SEI-89-TR-3, Software Engineering Institute (1989)
2. ISO/IEC FDIS 12207:2017 - Systems and software engineering - Software life cycle processes (2017)
3. ISO/IEC TR 29110-5-1-2:2011 - Software engineering Lifecycle profiles for VSEs: Management and engineering guide: Generic profile group: Basic profile (2011)
4. Wen, L., Tuffley, D., Rout, T.: Using composition trees to model and compare software process. In: O'Connor, R.V., Rout, T., McCaffery, F., Dorling, A. (eds.) SPICE 2011. CCIS, vol. 155, pp. 1–15. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21233-8_1
5. Golra, F.R., Dagnat, F., Bendraou, R., Beugnard, A.: Continuous process compliance using model driven engineering. In: Ouhammou, Y., Ivanovic, M., Abelló, A., Bellatreche, L. (eds.) MEDI 2017. LNCS, vol. 10563, pp. 42–56. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66854-3_4

[7] https://www.w3.org/TR/rdf-sparql-query/.

6. Boucher, Q., Perrouin, G., Deprez, J.-C., Heymans, P.: Towards configurable ISO/IEC 29110-compliant software development processes for very small entities. In: Winkler, D., O'Connor, R.V., Messnarz, R. (eds.) EuroSPI 2012. CCIS, vol. 301, pp. 169–180. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31199-4_15
7. Emmerich, W., Finkelstein, A., Montangero, C., Antonelli, S., Armitage, S., Stevens, R.: Managing standards compliance. IEEE Trans. Softw. Eng. **25**(6), 836–851 (1999)
8. Rout, T.P., El Emam, K., Fusani, M., Goldenson, D., Jung, H.-W.: SPICE inretrospect: developing a standard for process assessment. J. Syst. Softw. **80**, 1483–1493 (2007)
9. ISO/IEC 17000: Conformity assessment Vocabulary and general principles. International Organization for Standardization, Geneva (2004)
10. Varkoi, T.: Process assessment in very small entities-an ISO/IEC 29110 based method. In: 7th International Conference QUATIC. IEEE (2010)
11. ISO/IEC JTC1/SC7 WG10: Transition from ISO/IEC 15504 to ISO/IEC 330xx, Working Document (2017)
12. Proença, D., Borbinha, J.: A formalization of the ISO/IEC 15504: enabling automatic inference of capability levels. In: Mas, A., Mesquida, A., O'Connor, R.V., Rout, T., Dorling, A. (eds.) SPICE 2017. CCIS, vol. 770, pp. 197–210. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-67383-7_15
13. Wen, L., Rout, T.: Using composition trees to validate an entry profile of software engineering lifecycle profiles for very small entities (VSEs). In: Mas, A., Mesquida, A., Rout, T., O'Connor, R.V., Dorling, A. (eds.) SPICE 2012. CCIS, vol. 290, pp. 38–50. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-30439-2_4
14. Thaddeus, S., Kasmir Raja, K.: Ontology for software Engineering Process Automation (2006). Accessed http://www.researchgate.net/publication/278241783
15. Castellanos Ardila, J.P., Gallina, B.: Towards increased efficiency and confidence in process compliance. In: Stolfa, J., Stolfa, S., O'Connor, R.V., Messnarz, R. (eds.) EuroSPI 2017. CCIS, vol. 748, pp. 162–174. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-64218-5_13
16. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press, Cambridge (2003)
17. Wang, S., Jin, L., Jin, C.: Represent software process engneering metamodel in description logic. In: Proceedings of World Academy of Science, Engineering and Technology, vol. 11 (2006). ISSN 1307–6884
18. Diebold, P., Scherr, S.: Software process models vs. descriptions: what do practitioners use and need? J. Softw. Maint. Evol. Res. Pract. **29**, e1479 (2017)
19. Morales-Trujillo, M., Oktaba, H., Hernandedez-Quiroz, F., Escalante-Ramirez, B.: Towards a formalisation of a framework to express and reason about software engineering methods. Comput. Inform. **37**(1), 109–141 (2018)
20. Kabaale, E., Wen, L., Wang, Z., Rout, T.: Representing software process in description logics: an ontology approach for software process reasoning and verification. In: Clarke, P.M., O'Connor, R.V., Rout, T., Dorling, A. (eds.) SPICE 2016. CCIS, vol. 609, pp. 362–376. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-38980-6_26

21. Kabaale, E., Wen, L., Wang, Z., Rout, T.: An axiom based metamodel for software process formalisation: an ontology approach. In: Mas, A., Mesquida, A., O'Connor, R.V., Rout, T., Dorling, A. (eds.) SPICE 2017. CCIS, vol. 770, pp. 226–240. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-67383-7_17
22. ISO/IEC 15504–5:2012 - Information technology - Process assessment An exemplar Process Assessment Model, International Organization for Standardization and International Electrotechnical Commission Std. (2012)
23. ISO/IEC DIS 33020 - Information technology Process assessment Process measurement framework for assessment of proecess capability (2013)
24. Jung, H.W.: Investigating measurement scales and aggregation methods in SPICE assessment method. Inf. Softw. Technol. **55**(8), 1450–1461 (2013)
25. Guarino, N.: Formal ontology in information systems. In: Proceedings of FOIS98, Trento, Italy. IOS Press, Amsterdam (1998)
26. Motik, B., Horrocks, I., Sattler, U.: Bridging the gap between OWL and relational databases. In: WWW 2007 (2007)
27. Pellet: OWL 2 Reasoner for Java. http://clarkparsia.com/pellet/
28. Gonzalez-Perez, C., Henderson-Sellers, B., McBride, T., Low, G.C., Larrucea, X.: An ontology for ISO software engineering standards 2) proof of concept and application. Comput. Stand. Interfaces **48**, 112–123 (2016)
29. Tarhan, A., Giray, G.: On the use of ontologies in software process assessment: a systematic literature review. In: EASE (2017)
30. ISO/IEC TR 24774 - Software and systems engineering - Life cycle management - Guidelines for process description (2007)
31. Nash, E., Wiebensohn, J., Nikkil, R., Vatsanidou, A., Fountas, S., Bill, R.: Towards automated compliance checking based on a formal representation of agricultural production standards. Comput. Electron. Agric. **78**, 28–37 (2011)
32. Krishnamurthy, A., O'Connor, R.V.: Using ISO/IEC 12207 to analyze open source software development processes: an e-learning case study. In: Woronowicz, T., Rout, T., O'Connor, R.V., Dorling, A. (eds.) SPICE 2013. CCIS, vol. 349, pp. 107–119. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38833-0_10