AI-T: Software Testing Ontology for AI-based Systems

J. I. Olszewska
School of Computing and Engineering, University of West Scotland, U.K.

Keywords: Intelligent Systems, Software Testing, Software Engineering Ontology, Ontological Domain Analysis and Modeling, Knowledge Engineering, Knowledge Representation, Interoperability, Decision Support Systems, Transparency, Accountability, Unbiased Machine Learning, Explainable Artificial Intelligence (XAI).

Abstract: Software testing is an expanding area which presents an increasing complexity. Indeed, on one hand, there is the development of technologies such as Software Testing as a Service (TaaS), and on the other hand, there is a growing number of Artificial Intelligence (AI)-based softwares. Hence, this work is about the development of an ontological framework for AI-softwares' Testing (AI-T), which domain covers both software testing and explainable artificial intelligence; the goal being to produce an ontology which guides the testing of AI softwares, in an effective and interoperable way. For this purpose, AI-T ontology includes temporal interval logic modelling of the software testing process as well as ethical principle formalization and has been built using the Enterprise Ontology (EO) methodology. Our resulting AI-T ontology proposes both conceptual and implementation models and contains 708 terms and 706 axioms.

1 INTRODUCTION

Artificial Intelligence (AI) systems are nowadays pervasive in our Society, from face recognition (Raji et al., 2020) to cloud robotic systems (Pignaton de Freitas et al., 2020a). However, the black-box nature of many of these systems (Guidotti et al., 2018) leads to growing public concerns (Koene, 2017). Therefore, appropriate testing of such AI-based softwares is crucial to provide people and industry with dependable and ethical intelligent systems. Indeed, new-generation softwares are expected to be efficient, whilst transparent, safe, and secure (Olszewska, 2019b).

Software testing consists usually in a range of activities from manual testing to automated testing (Prasad, 2008), or Testing as a Service (TaaS) (Yu et al., 2010). They are applied throughout the software development life-cycle (SDLC) (Sommerville, 2015) and can be grouped into three processes, namely, the Organizational Test Process (OTP), the Test Management Process (TMP), and the Dynamic Test Process (DTP), which corresponding documentations report about (Alaqail and Ahmed, 2018).

Thus, OTP is utilized for the development and management of the organizational test specifications and includes three phases: to develop organizational test specification (OTP1); monitor and control the use of organizational test specifications (OTP2); and update organizational test specifications (OTP3). TMPs are used at the project level, and the three TMPs are defined as: test planning (TMP1); test monitoring and control (TMP2); and test completion (TMP3). On the other hand, the four DTPs comprise: test design & implementation (DTP1); test environment set-up & maintenance (DTP2); test execution (DTP3); and test incident reporting (DTP4). These DTPs are used to carry out dynamic testing within a particular level of testing (i.e. unit testing, integration testing, system testing, and acceptance testing) or type of testing (e.g. performance testing, security testing, usability testing) (IEEE, 2013a), (IEEE, 2013b), (IEEE, 2013c), (IEEE, 2015b), (IEEE, 2016).

In particular, ontologies are a convenient approach for capturing a conceptualization of the software testing domain (Tebes et al., 2019). Indeed, ontologies define terms, properties, relationships, and axioms explicitly, unambiguously, and in an interoperable way which could be understood by both humans and machines, since ontologies directly support automated reasoning about the domain knowledge representation (Chianese et al., 2009).

Hence, some ontologies have been developed for software testing (Ferreira de Souza et al., 2013a). These ontologies share common software testing knowledge, but differ among other, by their termino-

Olszewska, J.
AI-T: Software Testing Ontology for AI-based Systems.
DOI: 10.5220/0010147902910298
In *Proceedings of the 12th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2020) - Volume 2: KEOD*, pages 291-298
ISBN: 978-989-758-474-9

logical variations, since their terms are extracted from different body of knowledge, such as SoftWare Engineering Body of Knowledge (SWEBOK) (Freitas and Vieira, 2014), International Software Testing Qualifications Board (ISTQB) (Arnicans et al., 2013), and standards such as IEEE 610-1990 (Ferreira de Souza et al., 2013b), ISO/IEC 12207-1995 (Barbosa et al., 2006), ISO/IEC 9126-2001 (Cai et al., 2009), or IEEE 829-2008 (Vasanthapriyan et al., 2017).

Howbeit, many of these ontologies lack of axiomatization, e.g. (Guo et al., 2011), (Sapna and Mohanty, 2011), (Iliev et al., 2012), and some are rather mere taxonomies like (Cai et al., 2009) or (Anandaraj et al., 2011).

On the other hand, most of the software testing ontologies have a limited domain coverage, such as Web Service testing (Huo et al., 2003), Graphical User Interface (GUI) testing (Li et al., 2009), or Linux testing (Bezerra et al., 2018).

Moreover, no specific ontological methodologies have been applied to the development of ontologies such as Test Ontology Model (TOM) (Bai et al., 2008), Testing Maturing Model (TMM) ontology (Ryu et al., 2008), or Regression Test Execution (RTE) ontology (Campos et al., 2017).

These developed ontologies are useful, e.g. for determining dependability (Looker et al., 2005), probing interoperability (Yu et al., 2005), performing test reuse (Li and Zhang, 2012), test case generation (Crapo and Moitra, 2019), or test automation (Paydar and Kahani, 2010), but many of them do not have an implementation such as the Testing as a Service (TaaS) ontology (Yu et al., 2009).

Actually, very few software testing ontologies present both conceptual and implementation models, and these ontologies formalize a number of terms, e.g. 45 terms in the Reference Ontology on Software Testing (ROoST) (Ferreira de Souza et al., 2013b), 52 terms in OntoTest (Barbosa et al., 2006), 63 terms in the Software Testing Ontology for Web Service (STOWS) (Huo et al., 2003), and 194 terms in the Software Test Ontology (SwTO) (Bezerra et al., 2018).

Besides, ontologies have been designed for Artificial Intelligence (AI) domain (Bayat et al., 2016) and focus on intelligent agents (IA) (Valente et al., 2010), autonomous systems (AS) (Bermejo-Alonso et al., 2010), internet of things (IoT) (Ma et al., 2014), cyber-physical systems (CPS) (Dai et al., 2017), smart production systems (Terkaj and Urgo, 2014), or cloud robotic systems (CRS) (Pignaton de Freitas et al., 2020b).

However, no dedicated ontology has been developed for testing AI-based softwares.

In this paper, we propose a novel domain ontology for AI-softwares' Testing (AI-T).

Its core knowledge includes the software testing domain as well as the ethical artificial intelligence domain.

AI-T ontology has been coded in Web Ontology Language Descriptive Logic (OWL DL), which is considered as the international standard for expressing ontologies and data on the Semantic Web (Guo et al., 2007), and using the Protege tool in conjunction with the FaCT++ reasoner (Tsarkov, 2014).

The resulting expert system provides an interoperable solution to test AI-based softwares.

Thus, the contributions of this paper is manyfold. Indeed, as far as we know, we propose the first ontology which purpose is to aid the testing of AI-based softwares. For that, we introduce the software ethical principle testing concepts along with the mathematical formalization of the software testing process using temporal-interval descriptive logic (DL). Besides, it is the first time Enterprise Ontology (EO) methodology is applied to develop a software testing ontology.

The paper is structured as follows. Section 2 presents the purpose and the building of our ontology for AI-based-software Testing (AI-T), while its evaluation and documentation are described in Section 3. Conclusions are drawn up in Section 4.

2 PROPOSED AI-T ONTOLOGY

To develop the AI-T ontology, we followed the ontological development life cycle (Gomez-Perez et al., 2004) based on the Enterprise Ontology (EO) Methodology (Dietz and Mulder, 2020), since EO is well suited for software engineering applications (van Kervel et al., 2012).

The adopted ontological development methodology consists of four main phases (Olszewska and Allison, 2018), which cover the whole development cycle, as follows:

1. identifications of the purpose of the ontology (Section 2.1);
2. ontology building which consists of three parts: the capture to identify the domain concepts and their relations; the coding to represent the ontology in a formal language; and the integration to share ontology knowledge (Section 2.2);
3. evaluation of the ontology to check that the developed ontology meets the scope of the project (Section 3.1);
4. documentation of the ontology (Section 3.2).

Table 1: Artificial-Intelligence-based software' Testing (AI-T) Body of Knowledge (BoK) summary.

Sample Concepts	Terminology Sources
Software Testing Strategy	SWEBOK, IEEE 29119-1-2013
Software Testing Process	IEEE 29119-2-2013
Organizational Test Process	
Test Management Process	
Dynamic Test Process	
Software Testing Documentation	IEEE 29119-3-2013
Test Plan	
Software Testing Level	ISTQB
Unit Testing	
Integration Testing	
System Testing	
Acceptance Testing	
Software Testing Technique	IEEE 29119-4-2015, IEEE 24765-2017
Dynamic Testing	
Specification-Based Technique	
Structure-Based Technique	
Static Testing	
Software Testing Type	
Software Quality Testing	IEEE 29119-4-2015, (Sommerville, 2015)
Software Safety Testing	IEEE 1228-1994, (Leveson, 2020)
Software Ethical Principle Testing	IEEE EADv2-2019, VDEv1-2020, IEEE 70xx
Application-Based Testing (e.g. AI Software)	IEEE 1872-2015, (Leonard et al., 2017), (Olszewska, 2019a)
Software Comparison Testing	(Sommerville, 2015)
Software Testing Measure & Metric	(Pressman, 2010), (Olszewska, 2019b)

2.1 Ontology Purpose

The scope of this AI-T domain ontology is to assist testers with the testing of AI-based softwares. Indeed, software testing originates from professional practices that have led to a variety of testing approaches and generated different nomenclatures of the testing types and techniques. On the other hand, the testing of AI-based softwares is a new area without established models yet (Hutchinson and Mitchell, 2019). Therefore, the AI-T ontology aims to contribute to the elicitation of the Software Testing knowledge and the formalization of concepts for AI-based softwares' testing.

Since ontologies intrinsically allow to represent knowledge unambiguously, to share information in an interoperable way, and to perform automated reasoning, the AI-T ontology purpose is to contribute to (i) support human tester(s) in managing software testing; (ii) help human testers in testing AI-based softwares; (iii) guide intelligent agent(s) in generating/reusing test cases; (iv) facilitate intelligent agent(s)' learning about software testing; (v) aid collaborative mixed human-intelligent agent teams in testing software agents.

2.2 Ontology Building

The knowledge capture consists in the identification of concepts and relations of both Software Testing and Explainable Artificial Intelligence domains as summed up in Table 1.

Hence, AI-T body of knowledge for software engineering and in particular software testing comprises the SWEBOK guide (Bourque and Fairley, 2014), the ISTQB terminology (ISTQB, 2020), and software engineering standards such as IEEE 24765-2017 (IEEE, 2017) for the software engineering vocabulary, IEEE 29119-1-2013 (IEEE, 2013a) for software testing concepts and definitions, IEEE 29119-2-2013 (IEEE, 2013b) for the software testing test processes, IEEE 29119-3-2013 (IEEE, 2013c) for software testing test documentation, IEEE29119-4-2015 (IEEE, 2015b) for software testing test techniques, IEEE29119-5-2016 (IEEE, 2016) for software testing keyword-driven testing, (IEEE, 1994), (Leveson, 2020) for software safety testing as well as books such as (Sommerville, 2015) for software quality testing and (Pressman, 2010) for software measures and metrics.

On the other hand, AI-T body of knowledge for ethical AI general principles (i.e. human rights, well-being, accountability, transparency, awareness

of misuse) includes the IEEE Ethically Aligned Design guidelines (EADv2) (IEEE, 2019), the interdisciplinary framework for AI ethical practice (e.g. transparency, accountability, privacy, justice, reliability, environmental sustainability) (Hallensleben and Hustedt, 2020), the ethical standards in robotics and AI (Koene et al., 2018b), (Winfield, 2019), (Bryson and Winfield, 2017), (Olszewska et al., 2018), (Olszewska et al., 2020) and subsequent standards such as IEEE P7000 (Spiekermann, 2017), IEEE P7001 for transparency (Wortham et al., 2016), IEEE P7003 (Koene et al., 2018a) for accountability, or IEEEE P7010 (IEEE, 2020) for well-being measurement.

The knowledge coding is done in Descriptive Logic (DL) and uses temporal-interval logic relations as introduced in (Olszewska, 2016). Thence, within the Software Testing Process which has been described in Section 1, the concept of 'Organizational Test Process (OTP)' is defined in DL, as follows:

$$\begin{aligned} Organizational\_Test\_Process \sqsubseteq Software\_Testing\_Process \\ \sqcap \exists hasProcess_{=\{OTP_1=Develop\_OTS_n\}} \\ \sqcap \exists hasProcess_{=\{OTP_2=Monitor\_OTS_n\}} \\ \sqcap \exists hasProcess_{=\{OTP_3=Update\_OTS_n\}}, \end{aligned} \quad (1)$$

with OTS_n, an organizational test specification.

Furthermore, the OTP processes could be formalised in temporal DL, as follows:

$$\begin{aligned} OTP\_Implementation \sqsubseteq Organizational\_Test\_Process \\ \sqcap (\diamond t_1 ... t_3) \\ (OTP_1 m OTP_2) ... (OTP_1 < OTP_3) \\ \cdot (OTP_1 @ t_1 \sqcap ... \sqcap OTP_3 @ t_3), \end{aligned} \quad (2)$$

with $before$ and $meet$, the temporal-interval relations as defined respectively in temporal DL:

$$\begin{aligned} P_i < P_j \equiv before(P_i @ t_i, P_j @ t_j) \sqsubseteq Temporal\_Relation \\ \sqcap (\diamond t_i)(\diamond t_j) \\ (t_{i^+} < t_{j^-}) \\ \cdot (P_i @ t_i \sqcap P_j @ t_j), \end{aligned} \quad (3)$$

$$\begin{aligned} P_i m P_j \equiv meet(P_i @ t_i, P_j @ t_j) \sqsubseteq Temporal\_Relation \\ \sqcap (\diamond t_i)(\diamond t_j) \\ (t_{i^+} = t_{j^-}) \\ \cdot (P_i @ t_i \sqcap P_j @ t_j), \end{aligned} \quad (4)$$

where the temporal DL symbol $\diamond$ represents the temporal existential qualifier, and where a time interval is an ordered set of points $T = \{t\}$ defined by end-points t^- and t^+, such as $(t^-, t^+) : (\forall t \in T)(t > t^-) \wedge (t < t^+)$.

The concept of 'Test Management Process (TMP)' could be represented in DL, as follows:

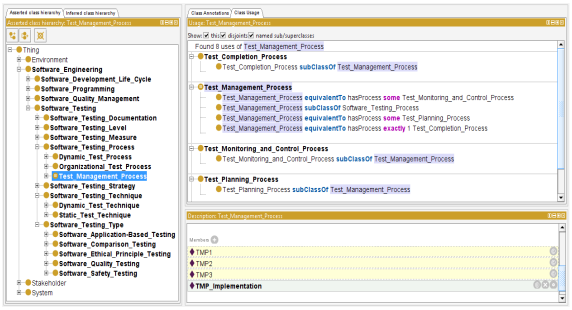

Figure 1: Main classes of the Software Testing domain.

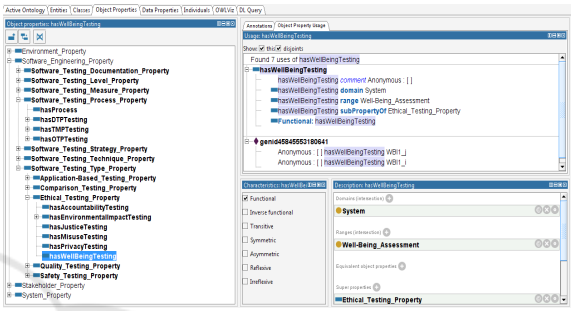

Figure 2: Main properties of the Ethical AI domain.

$$\begin{aligned} Test\_Management\_Process \sqsubseteq Software\_Testing\_Process \\ \sqcap \exists hasProcess_{=\{TMP_1=Test\_Planning\}} \\ \sqcap \exists hasProcess_{=\{TMP_2=Test\_Monitoring\_and\_Control\}} \\ \sqcap \exists hasProcess_{=\{TMP_3=Test\_Completion\}}. \end{aligned} \quad (5)$$

Moreover, the TMP processes could be formalised in temporal DL, as follows:

$$\begin{aligned} TMP\_Implementation \sqsubseteq Test\_Management\_Process \\ \sqcap (\diamond t_1 ... t_3) \\ (TMP_1 < TMP_3) ... (TMP_2 < TMP_3) \\ \cdot (TMP_1 @ t_1 \sqcap ... \sqcap TMP_3 @ t_3). \end{aligned} \quad (6)$$

The concept of 'Dynamic Test Process (DTP)' could be represented in DL, as follows:

$$\begin{aligned} Dynamic\_Test\_Process \sqsubseteq Software\_Testing\_Process \\ \sqcap \exists hasProcess_{=\{DTP_1=Test\_Design\_and\_Implementation\}} \\ \sqcap \exists hasProcess_{=\{DTP_2=Test\_Environment\_Setup\_and\_Maintenance\}} \\ \sqcap \exists hasProcess_{=\{DTP_3=Test\_Execution\}} \\ \sqcap \exists hasProcess_{=\{DTP_4=Test\_Incident\_Reporting\}}. \end{aligned} \quad (7)$$

The DTP processes could be also formalised in temporal DL, as follows:

$$
\begin{aligned}
DTP\_Implementation \sqsubseteq\ & Dynamic\_Test\_Process \\
& \sqcap (\diamond t_1 \ldots t_4) \\
& (DTP_1 mDTP_2)(DTP_1 mDTP_3)(DTP_2 mDTP_3) \\
& (DTP_3 < DTP_4) \\
& \cdot (DTP_1 @ t_1 \sqcap \ldots \sqcap DTP_4 @ t_4).
\end{aligned} \tag{8}
$$

It is worth noting that the temporal-interval relations *before* and *meet* in Eq. 6 and Eq. 8 are as defined in Eqs. 3-4, respectively.

On the other hand, ethical AI properties such as *hasWellBeingTesting* could be formalized in DL, as follows:

$$
\begin{aligned}
hasWellBeingTesting \sqsubseteq\ & Ethical\_Testing\_Property \\
& \sqcap \exists hasWellBeing_{=\{WBI1_i\}} \\
& \sqcap \exists hasWellBeing_{=\{WBI1_j\}} \\
& \sqcap \exists hasWellBeingValue_{=\{WBI1_j\} \geq \{WBI1_i\}}.
\end{aligned} \tag{9}
$$

with $WBI1_i$, a well-being indicator before using the AI-based software and $WBI1_j$, this well-being indicator after using the AI-based software.

The ontology is implemented in the Web Ontology Language (OWL) language, which is the language of all the software testing ontologies (Ferreira de Souza et al., 2013a), and uses Protege v4.0.2 Integrated Development Environment (IDE) with the inbuilt FaCT++ v1.3.0 reasoner (Tsarkov, 2014) to check the internal consistency and to perform automated reasoning on the terms and axioms. An excerpt of the encoded concepts is presented in Fig. 1, while some properties are shown in Fig. 2.

3 VALIDATION AND DISCUSSION

The developed AI-T ontology has been evaluated both quantitatively and qualitatively in a series of experiments as described in Sections 3.1, while its documentation is mentioned in 3.2.

3.1 Ontology Evaluation

The ontology evaluation ensures that the developed ontology meets all the requirements (Dobson et al., 2005).

For this purpose, experiments have been carried out using Protege v4.0.2 IDE and applying FaCT++ v1.3.0 reasoner. In particular, an example including the mathematical model and OWL implementation has been provided for testers to get automated guidance during the management process through running

Metrics

Class count	708
Object property count	87
Data property count	8
Individual count	37
DL expressivity	SROIQ(D)

Figure 3: Some values of the AI-T ontology metrics.

AI-T ontology (Fig. 1). On the other hand, an example of testers being supported in testing AI-based softwares, e.g. for the well-being assessment, has been formalized in Section 2.2 and illustrated in Fig. 2, leading to patterns for reusing test cases and facilitating the software testing learning and interoperable knowledge sharing. In qll these experiments, AI-T ontology provided 100% correct answers, and no inconsistency has been observed.

The evaluation of AI-T used metrics such as presented in (Tartir et al., 2018) and (Hlomani and Stacey, 2014). The computed values by Protege are presented in Fig. 3. We can observe that the DL expressivity is $\mathcal{SROIQ}(\mathcal{D})$. It is worth noting the reasoner works under the open-world assumption, i.e. if for a question, there is no information in the ontology, then the answer of the system is 'does not know', and not 'does not exist'. To obtain the latter one, information should be explicitly provided, but adding all these closure-type assertions can slow down the reasoner. So, in practice, a trade-off should be achieved between computational efficiency and completeness. Actually, AI-T performs in real time and contains a total of 707 terms and 705 axioms. This is much higher compared to other state-of-art software testing ontologies that own in average 61 ± 44 terms (Tebes et al., 2019). Moreover, AI-T cohesion could be assessed using the number of root classes which is equal to 1, the number of leaf classes which is equal to 521, and the average depth which is equal to 4. All these measures indicate AI-T shows promising performance for real-world deployment.

3.2 Ontology Documentation

The AI-T ontology has been documented in Section 2. It is a middle-out, domain ontology which has been developed from scratch using non-ontological resources such as primary sources, e.g. IEEE standards, as recapped in Table 1. AI-T is not dependent of any particular software/system/agent/service/application/project, but it is rather focused on the software testing knowledge as well as the AI system's ethical principles.

It is worth noting the AI-T has not reused any existing software testing ontology, and it is the first ontology in its kind for the AI-softwares' testing domain. Moreover, the AI-T ontology could be used in conjunction with other ontologies such as the core ontology for autonomous systems (CORA) (IEEE, 2015a) or other robotics and automation ontologies (Fiorini et al., 2017) for further integration.

4 CONCLUSIONS

With the increasing usage of artificial intelligence (AI) and especially machine learning (ML) within softwares as well as the growing number of autonomous intelligent agents actioning outside systems, the effective testing of AI-based softwares is crucial and should have both a wide test coverage and a rigorous formalization. Hence, we propose an ontological approach for testing AI-based softwares. In particular, we proposed a new domain ontology which is called AI-T and which encompasses software testing knowledge and ethical AI principles. AI-T purpose is to guide AI-based softwares' testing, to automate their testing, and also facilitate the learning about software testing. AI-T ontology has both conceptual and implementation models that are helpful for the interoperable sharing of the AI-based software testing terms and relations, and that have been successfully applied for whatever manual or automated testing.

REFERENCES

Alaqail, H. and Ahmed, S. (2018). Overview of software testing standard ISO/IEC/IEEE 29119. *International Journal of Computer Science and Network Security*, 18(2):112–116.

Anandaraj, A., Kalaivani, P., and Rameshkumar, V. (2011). Development of ontology-based intelligent system for software testing. *International Journal of Communication, Computation and Innovation*, 2(2):157–161.

Arnicans, G., Romans, D., and Straujums, U. (2013). Semi-automatic generation of a software testing lightweight ontology from a glossary based on the ONTO6 methodology. *Frontiers in Artificial Intelligence and Applications*, pages 263–276.

Bai, X., Lee, S., Tsai, W.-T., and Chen, Y. (2008). Ontology-based test modeling and partition testing of web services. In *Proceedings of IEEE International Conference on Web Services*, pages 465–472.

Barbosa, E. F., Nakagawa, E. Y., and Maldonado, J. C. (2006). Towards the establishment of an ontology of software testing. In *Proceedings of IEEE International Conference on Software Engineering and Knowledge Engineering (SEKE)*, pages 522–525.

Bayat, B., Bermejo-Alonso, J., Carbonera, J. L., Facchinetti, T., Fiorini, S. R., Goncalves, P., Jorge, V., Habib, M., Khamis, A., Melo, K., Nguyen, B., Olszewska, J. I., Paull, L., Prestes, E., Ragavan, S. V., Saeedi, S., Sanz, R., Seto, M., Spencer, B., Trentini, M., Vosughi, A., and Li, H. (2016). Requirements for building an ontology for autonomous robots. *Industrial Robot*, 43(5):469–480.

Bermejo-Alonso, J., Sanz, R., Rodriguez, M., and Hernandez, C. (2010). Ontology-based engineering of autonomous systems. In *Proceedings of IEEE International Conference on Autonomic and Autonomous Systems*, pages 47–51.

Bezerra, D., Costa, A., and Okada, K. (2018). SWTOI (software test ontology integrated) and its application in Linux test. In *Proceedings of IEEE International Workshop on Ontology, Conceptualization and Epistemology for Information Systems, Software Engineering and Service Science*, pages 25–36.

Bourque, P. and Fairley, R. E. (2014). SWEBOK: Guide to Software Engineering Body of Knowledge.

Bryson, J. and Winfield, A. (2017). Standardizing ethical design for artificial intelligence and autonomous systems. *Computer*, 50(5):116–119.

Cai, L., Tong, W., Liu, Z., and Zhang, J. (2009). Test case reuse based on ontology. In *Proceedings of IEEE Pacific Rim International Symposium on Dependable Computing*, pages 103–108.

Campos, H., Acacio, C., Braga, R., Araujo, M. A. P., David, J. M. N., and Campos, F. (2017). Regression tests provenance data in the continuous software engineering context. In *Proceedings of IEEE Brazilian Symposium on Systematic and Automated Software Testing*, pages 1–6.

Chianese, A., Fasolino, A., Moscato, V., and Tramontana, P. (2009). Using ontologies to achieve semantic interoperability in the Web: An approach based on the semantic triangle model. In *Proceedings of Joint Working IEEE International Conference on Intelligent Systems Design and Applications*, pages 884–889.

Crapo, A. W. and Moitra, A. (2019). Using OWL ontologies as a domain-specific language for capturing requirements for formal analysis and test case generation. In *Proceedings of IEEE International Conference on Semantic Computing*, pages 361–366.

Dai, W., Dubinin, V. N., Christensen, J. H., Vyatkin, V., and Guan, X. (2017). Toward self-manageable and adaptive industrial cyber-physical systems with knowledge-driven autonomic service management. *IEEE Transactions on Industrial Informatics*, 13(2):725–736.

Dietz, J. and Mulder, H. (2020). *Enterprise Ontology*. Springer.

Dobson, G., Lock, R., and Sommerville, I. (2005). QoSOnt: A QoS ontology for service-centric systems. In *Proceedings of the EUROMICRO International Conference on Software Engineering and Advanced Applications*, pages 80–87.

Ferreira de Souza, E., de Almeida Falbo, R., and Vijaykumar, N. L. (2013a). Ontologies in software testing: A systematic literature review. In *Proceedings of the Seminar on Ontology Research in Brazil*, pages 71–82.

Ferreira de Souza, E., de Almeida Falbo, R., and Vijaykumar, N. L. (2013b). Using ontology patterns for building a reference software testing ontology. In *Proceedings of IEEE International Enterprise Distributed Object Computing Conference Workshops*, pages 21–30.

Fiorini, S. R., Bermejo-Alonso, J., Goncalves, P., Pignaton de Freitas, E., Olivares Alarcos, A., Olszewska, J. I., Prestes, E., Schlenoff, C., Ragavan, S. V., Redfield, S., Spencer, B., and Li, H. (2017). A suite of ontologies for robotics and automation. *IEEE Robotics and Automation Magazine*, 24(1):8–11.

Freitas, A. and Vieira, R. (2014). An ontology for guiding performance testing. In *Proceedings of IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, pages 400–407.

Gomez-Perez, A., Fernandez-Lopez, M., and Corcho, O. (2004). *Ontological Engineering*. Springer-Verlag.

Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., and Pedreschi, D. (2018). A survey of methods for explaining black box models. *ACM Computing Surveys*, 51(5):1–42.

Guo, S., Zhang, J., Tong, W., and Liu, Z. (2011). An application of ontology to test case reuse. In *Proceedings of IEEE International Conference on Mechatronic Science, Electric Engineering and Computer*, pages 775–778.

Guo, Y., Qasem, A., Pan, Z., and Heflin, J. (2007). A requirement driven framework for benchmarking semantic web knowledge base systems. *IEEE Transactions on Knowledge and Data Engineering*, 19(2):297–309.

Hallensleben, S. and Hustedt, C. (2020). AI Ethics Impact Group. From Principles to Practice. An Interdisciplinary Framework to Operationalise AI Ethics.

Hlomani, H. and Stacey, D. (2014). Approaches, methods, metrics, measures, and subjectivity in ontology evaluation: A survey. *Semantic Web Journal*, 1(5):1–11.

Huo, Q., Zhu, H., and Greenwood, S. (2003). A multiagent software engineering environment for testing Web-based applications. In *Proceedings of IEEE International Computer Software and Applications Conference*, pages 210–215.

Hutchinson, B. and Mitchell, M. (2019). 50 years of test (un) fairness: Lessons for machine learning. In *Proceedings of ACM Conference on Fairness, Accountability, and Transparency*, pages 49–58.

IEEE (1994). IEEE Standard for Software Safety Plans: IEEE 1228-1994.

IEEE (2013a). ISO/IEC/IEEE International Standard - Software and Systems Engineering - Software Testing - Part 1: Concepts and Definitions: IEEE 29119-1-2013.

IEEE (2013b). ISO/IEC/IEEE International Standard - Software and Systems Engineering - Software Testing - Part 2: Test Processes: IEEE 29119-2-2013.

IEEE (2013c). ISO/IEC/IEEE International Standard - Software and Systems Engineering - Software Testing - Part 3: Test Documentation: IEEE 29119-3-2013.

IEEE (2015a). IEEE Standard Ontologies for Robotics and Automation: IEEE 1872-2015.

IEEE (2015b). ISO/IEC/IEEE International Standard - Software and Systems Engineering - Software Testing - Part 4: Test Techniques: IEEE 29119-4-2015.

IEEE (2016). ISO/IEC/IEEE International Standard - Software and Systems Engineering - Software Testing - Part 5: Keyword-Driven Testing: IEEE 29119-5-2016.

IEEE (2017). ISO/IEC/IEEE International Standard - Systems and Software Engineering - Vocabulary: IEEE 24765-2017.

IEEE (2019). IEEE Standard Association. Ethically Aligned Design.

IEEE (2020). IEEE Recommended Practice for Assessing the Impact of Autonomous and Intelligent Systems on Human Well-Being: IEEE 7010-2020.

Iliev, M., Karasneh, B., Chaudron, M. R. V., and Essenius, E. (2012). Automated prediction of defect severity based on codifying design knowledge using ontologies. In *Proceedings of IEEE International Workshop on Realizing AI Synergies in Software Engineering*, pages 7–11.

ISTQB (2020). International software testing qualifications board.

Koene, A. (2017). Algorithmic bias: Addressing growing concerns. *IEEE Technology and Society Magazine*, 36(2):31–32.

Koene, A., Dowthwaite, L., and Seth, S. (2018a). IEEE P7003 Standard for algorithmic bias considerations. In *Proceedings of IEEE/ACM International Workshop on Software Fairness*, pages 38–41.

Koene, A., Smith, A. L., Egawa, T., Mandalh, S., and Hatada, Y. (2018b). IEEE P70xx, Establishing standards for ethical technology. In *Proceedings of ACM Conference on Knowledge Discovery and Data Mining*, pages 1–2.

Leonard, S., Allison, I. K., and Olszewska, J. I. (2017). Design and test (D&T) of an in-flight entertainment system with camera modification. In *Proceedings of the IEEE International Conference on Intelligent Engineering Systems*, pages 151–156.

Leveson, N. G. (2020). Are you sure your software will not kill anyone? *Communications of the ACM*, 63(2):25–28.

Li, H., Chen, F., Yang, H., Guo, H., Chu, W. C.-C., and Yang, Y. (2009). An ontology-based approach for GUI testing. In *Proceedings of IEEE International Computer Software and Applications Conference*, pages 632–633.

Li, X. and Zhang, W. (2012). Ontology-based testing platform for reusing. In *Proceedings of IEEE International Conference on Internet Computing for Science and Engineering*, pages 86–89.

Looker, N., Gwynne, B., Xu, J., and Munro, M. (2005). An ontology-based approach for determining the dependability of service-oriented architectures. In *Proceedings of Annual IEEE International Workshop on Object-Oriented Real-Time Dependable Systems*, pages 1–8.

Ma, M., Wang, P., and Chu, C.-H. (2014). Ontology-based semantic modeling and evaluation for Internet of Things applications. In *Proceedings of IEEE International Conference on Internet of Things*, pages 24–30.

Olszewska, J. I. (2016). Temporal interval modeling for UML activity diagrams. In *Proceedings of the International Conference on Knowledge Engineering and Ontology Development (KEOD)*, pages 199–203.

Olszewska, J. I. (2019a). D7-R4: Software development life-cycle for intelligent vision systems. In *Proceedings of the International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (KEOD)*, pages 435–441.

Olszewska, J. I. (2019b). Designing transparent and autonomous intelligent vision systems. In *Proceedings of the International Conference on Agents and Artificial Intelligence*, pages 850–856.

Olszewska, J. I. and Allison, I. K. (2018). ODYSSEY: Software development life cycle ontology. In *Proceedings of the International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (KEOD)*, pages 303–311.

Olszewska, J. I., Houghtaling, M., Goncalves, P., Haidegger, T., Fabiano, N., Carbonera, J. L., Fiorini, S. R., and Prestes, E. (2018). Robotic ontological standard development life cycle. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–6.

Olszewska, J. I., Houghtaling, M., Goncalves, P. J. S., Fabiano, N., Haidegger, T., Carbonera, J. L., Patterson, W. R., Ragavan, S. V., Fiorini, S. R., and Prestes, E. (2020). Robotic standard development life cycle in action. *Journal of Intelligent and Robotic Systems*, 98:119–131.

Paydar, S. and Kahani, M. (2010). Ontology-based web application testing. In *Proceedings of IEEE Conference on Novel Algorithms and Techniques in Telecommunications and Networking*, pages 23–27.

Pignaton de Freitas, E., Bermejo-Alonso, J., Khamis, A., Li, H., and Olszewska, J. I. (2020a). Ontologies for cloud robotics. *Knowledge Engineering Review*, 35:1–17.

Pignaton de Freitas, E., Olszewska, J. I., Carbonera, J. L., Fiorini, S., Khamis, A., Sampath Kumar, V. R., Barreto, M., Prestes, E., Habib, M., Redfield, S., Chibani, A., Goncalves, P., Bermejo-Alonso, J., Sanz, R., Tosello, E., Olivares Alarcos, A., Konzen, A. A., Quintas, J., and Li, H. (2020b). Ontological concepts for information sharing in cloud robotics. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–14.

Prasad, K. V. K. K. (2008). *ISTQB Certification Study Guide*. Wiley.

Pressman, R. S. (2010). Product metrics. In *Software engineering: A practitioner's approach*, pages 613–643. McGraw-Hill, 7th edition.

Raji, I. D., Gebru, T., Mitchell, M., Buolamwini, J., Lee, J., and Denton, E. (2020). Saving face: Investigating the ethical concerns of facial recognition auditing. In *Proceedings of AAAI/ACM Conference on AI, Ethics, and Society*, pages 145–151.

Ryu, H., Ryu, D.-K., and Baik, J. (2008). A strategic test process improvement approach using an ontological description for MND-TMM. In *Proceedings of IEEE/ACIS International Conference on Computer and Information Science*, pages 561–566.

Sapna, P. G. and Mohanty, H. (2011). An ontology based approach for test scenario management. In *Proceedings of International Conference on Information Intelligence, Systems, Technology and Management*, pages 91–100.

Sommerville, I. (2015). *Software Engineering*. Pearson, 10th edition.

Spiekermann, S. (2017). IEEE P7000 - The first global standard process for addressing ethical concerns in system design. In *Proceedings of IS4SI Summit on Digitalisation for a Sustainable Society*, page 159.

Tartir, S., Arpinar, I. B., Moore, M., Sheth, A. P., and Aleman-Meza, B. (2018). OntoQA: Metric-based ontology quality analysis. In *IEEE International Conference on Data Mining Workshop*, pages 559–564.

Tebes, G., Peppino, D., Becker, P., Matturro, G., Solari, M., and Olsina, L. (2019). A systematic review on software testing ontologies. In *Proceedings of International Conference on the Quality of Information and Communications Technology*, pages 144–160.

Terkaj, W. and Urgo, M. (2014). Ontology-based modeling of production systems for design and performance evaluation. In *Proceedings of IEEE International Conference on Industrial Informatics (INDIN)*, pages 748–753.

Tsarkov, D. (2014). Incremental and persistent reasoning in FaCT++. In *Proceedings of OWL Reasoner Evaluation Workshop (ORE)*, pages 16–22.

Valente, P., Hossain, S., Gronbaek, B., Hallenborg, K., and Reis, L. P. (2010). A multi-agent framework for coordination of intelligent assistive technologies. In *Proceedings of the Iberian Conference on Information Systems and Technologies*, pages 1–6.

van Kervel, S., Dietz, J., Hintzen, J., van Meeuwen, T., and Zijlstra, B. (2012). Enterprise ontology driven software engineering. In *Proceedings of the International Conference on Software Technologies (ICSOFT)*, pages 205–210.

Vasanthapriyan, S., Tian, J., Zhao, D., Xiong, S., and Xiang, J. (2017). An ontology-based knowledge sharing portal for software testing. In *Proceedings of IEEE International Conference on Software Quality, Reliability and Security Companion*, pages 472–479.

Winfield, A. (2019). Ethical standards in robotics and AI. *Nature Electronics*, 2(2):46–48.

Wortham, R. H., Theodorou, A., and Bryson, J. J. (2016). What does the robot think? Transparency as a fundamental design requirement for intelligent systems. In *Proceedings of AAAI International Joint Conference on Artificial Intelligence Workshop*, pages 1–7.

Yu, L., Tsai, W.-T., Chen, X., Liu, L., Zhao, Y., Tang, L., and Zhao, W. (2010). Testing as a Service over cloud. In *Proceedings of IEEE International Symposium on Service Oriented System Engineering*, pages 181–188.

Yu, L., Zhang, L., Xiang, H., Su, Y., Zhao, W., and Zhu, J. (2009). A framework of Testing as a Service. In *Proceedings of IEEE International Conference on Management and Service Science*, pages 1–4.

Yu, Y., Huang, N., and Ye, M. (2005). Web services interoperability testing based on ontology. In *Proceedings of IEEE International Conference on Computer and Information Technology*, pages 1–5.