KARB Solution:
Compliance to Quality by Rule Based Benchmarking

Mohammad Reza Besharati
Mohammad Izadi
1- PhD Candidate, Sharif University of Technology, Tehran, Iran.
2- Assistant Professor, Sharif University of Technology, Tehran, Iran.

Introduction

Instead of proofs or logical evaluations, compliance assessment could be done by benchmarking. Benchmarks, in their nature, are applied. So a set of benchmarks could shape an applied solution for compliance assessment. In this paper, we introduce the KARB solution: Keeping away compliance Anomalies by Rule-based Benchmarking. By rule-based benchmarking, we mean evaluation of under-compliance-system by its symbolic specification and by using a set of symbolic rules (on behalf of semantic logic of evaluation).

Many Other approaches to compliance assessment have a theoretical nature: they are based on a systematic, rigorous or formal theory of proof or evaluation (Logics, formal Languages, reference models, domain models, etc.). Theories are in a high level of internal-integrity but they suffer from a lack of support and adaptation to diversity and complexity of real-world cases. In some situations, an aggregation of multiple simpler tools is more successful and more useful than a single, rigid, unified, in-depth-designed and sophisticated one.

More lightweight approaches to formal specification which support semantic modeling (like generative grammars, production rules, set-theoretic notations, rewriting logics, etc.) could play a key role in overcoming the challenge of semantic diversity and complexity of real-world compliance scenarios. These approaches support a kind of semantic compilation of several different, diverse and domain-specific semantic models. For example, by a set of generative rules, we could define the mapping and composition logic of different and independent semantic models.

About Compliance Checking

Compliance solutions are about assessment, evaluation, verification, validation or checking of a system, service, process, product, design, organization or environment regards rules, regulations, laws, standards, specifications, policies, guidelines, protocols, methods, principals, reference-models, etc. Many application domains benefit and need compliance solutions, including organizations and corporates in these domains: software and IT industry, E-Governance, finance and banking, legal section and professions, commerce and trade, highly regulated industries (such as food and drug, medical services and devices, construction industry, etc.), Complex and Interdisciplinary products and services, emerging
technology products and services (such as cyber-physical systems, self-driving cars, cognitive robotics and assistants, smart applications, etc.).

There are famous compliance concerns that have been considered by numerous regulations, standards, laws, and acts. Concerns such as Security, safety, privacy, protection, accountability, responsibility, transparency, competency, anti-piracy, anti-corruption, antitrust, accessibility, HCI, quality management and assurance, environmental management, sustainability, usability, human comfort, ethics, disability, children and elder people adherence, simplicity and ease of use are among them.

Modern Paradigms have amplified the necessity for compliance requirements (paradigms such as standardization in business, automation in industries, artificial intelligence and Ubiquitous computing in society, complex systems engineering, socio-technical systems, continuous growth in the economy and social complexity, quality maturity of services, processes, etc.)

In [21], a formal definition (as a 4-tuple) for a special kind of benchmarking is provided. There are formally defined frameworks for compliance checking in legal applications which defined in a set-theoretic manner, formal system [59] and Conceptual Modeling of legal texts [80]. Grammar-like and production-rule formalisms have been suggested for automated compliance checking in legal applications [55], [56], [57], [58].

Rules and grammars for architecture conformance checking, especially for assurance of “engineering quality” of the software [73]. Some rule-based approaches for architecture selection have been proposed to map the non-functional requirements, domain requirements and quality characteristics to architecture styles, architecture models and architectural patterns (for example see [79]).

Circuits and flows are very recurrent modeling approaches in systems engineering. Some researchers consider circuits and flows as a basis for compliance modeling, checking and benchmarking [70], [71]. There are some numerous verification tools and solutions for flow-based models that also could be served as a means of compliance checking. For example, agent-coordination protocols for crisis situations could be modeled and checked by these tools and solutions [86].

In software engineering, there are some model-based approaches for compliance assurance [76]. These approaches use a modeling notation or framework (such as UML, KAOS, GSN, etc.). “Model based assurance cases” is an approach for safety compliance management. It encompasses a compliance metamodel which contains “claim” or “requirement”, “evidence”, “arguments” and “context” [74], [75]. In [76], the need for a general model of compliance and compliance activity is addressed as a yet open-problem.

In [69], [72], a very close approach is introduced. The meta-model and system architecture of their approach is comparable with ours. There are some other meta-models for compliance checking applications and frameworks (for example see [60], [61], [62], [63]).

**About Benchmarking**

A benchmark is a common or standard infrastructure to examine, evaluate and compare the reality of solutions, tools or systems by their execution (for some definitions see [20], [4], [10], [11], [16], [19], [22] and [24]. For some examples in other fields see [5], [6], [7], [8] and [9]). Sometimes, simply a measure could be served as a benchmark [68].
For some early attempts in the history of benchmarking in IT and computing see [39] to [43] references. Also, there was a hot trend for benchmarking in quality assurance, management and continues process improvement [50], [32], which is a progressive line till now [51], [53], [52].

![Figure 1- a segmented word cloud to show the emphasis on different concepts in benchmark definitions.](image)

From a managerial point of view, benchmarking requires a significant investment in time and perhaps money [32]. So it should be considered as a long-term profitable activity and a sort of infrastructure building for a field. For example, in the field of cloud computing, the prior investments in performance measuring tools resulted in already available tools for the new filed (for a case, see [33]); it was a chance but also dedicated attempts began for defining and developing from-scratch benchmarks for cloud computing [34]. Form a managerial point of view, Investment in benchmarking is important [54] and this decision can give benefits to all stakeholders [51].

Successful notions of benchmarking (in each field) have a community that creates, promotes and uses the benchmarks. Benchmarking could be viewed as an applied manifestation and adoption of communities’ knowledge and expertise [1].

There are informal guidelines for ensuring the quality of developed software (for quality attributes such as security, integrity, maintainability, etc.). Formal specification and automatic checking of these guidelines could contribute to better quality assurance of software (see [67] as an example for security guidelines formalization and automation).

Benchmarks can assess the quality (rather than only functionality) [2], [19], [24], so they are suitable for formal or systematic qualitative analysis of systems. For example, security and compliance benchmarks have been reported [11], [15]. Measuring Productivity of an organization is another example that has qualitative dimensions (such as “the level of customer satisfaction, the quality of the product or the extent to which an organization has the right set of staff” [51]) which could be measured by some systematic approaches [51].

**Soft Benchmarks**

We could create a knowledge representation once and use it many times (Ontologies are a practical example of this manner of reusability. for example, see [23]). So a community could construct a knowledge representation and serve it as a common and reusable asset. If this asset helps the community to share their expertise, examine their systems and solutions and evaluate the behavior and other realities of their
systems, then we could name it “a soft benchmark”. The “soft” part of its name is added to indicate its knowledge-related nature.

Knowledge representations are not limited to ontologies [30]. Formal specifications, such as logical formulations, description logics, semantic networks, and rule-based approaches, are alternatives [28]. Logical models have a share in the approaches of compliance checking. For example, Logical modeling of regulations is a method for rule-representation and checking-automation [78] or using Rules as a paradigm for knowledge representation [31].

A logical theory for a piece of knowledge has the three required characteristics for a benchmark: 1) it could be a common infrastructure because of the reusable and defined nature of a formal specification, 2) the results of reasoning indicate an examination and evaluation of the system-under-study and provide a basis for comparison between alternative and competing systems, 3) executing a reasoning on a logical theory of a piece of knowledge is an execution for the meanings and semantics behind that knowledge. Thinking and mental activities could execute a hypothetical situation. In KARB, rule-based reasoning schema could be viewed as a mimic of these natural procedures (simulation of human auditing by automated and intelligent compliance audit tools is a proper need for compliance industry [13], [11], [12]).

Object models could be served as a semantic model [64], especially also for compliance checking proposes [65], [66]. For example, Fornax objects capture specific rule semantics for compliance checking of building designs [65]. These object models are context, domain and sometimes system-specific [64]. In the case of Fornax, the objects for hospital design semantics are different from those for airport design [65].

In KARB, knowledge is represented through an intuitionistic formal semantics which we call it “semantic logic”. Semantic logic was created to capture the semantic and meaning of text [0]. We use it in KARB for knowledge representation. Any knowledge has its semantic and meaning [25], [26], [27], [28]. If we capture its semantic and meaning, the knowledge itself is captured. Knowledge also has structure [29]. So a meaning structure (or semantic construction) could be a proper candidate for the manifestation of knowledge.

Based on and Adopted from [0], we consider semantic and meaning as a construction, lattice\(^1\) or system of realities (=intuitions). Any well-meaning formulation of knowledge represents and references to a combination of entities, things, objects, events, affairs, facts, physics, concepts, cognitions, affections or any other sort of basic realities and intuitions. So we could aggregately and abstractly consider knowledge as a combination and construction of basic realities and intuitions (with a glue of operators such as logical, structural, modalities and any needed one). This manner of semantic definition is a constructive and intuitionistic one.

A set of generative rules on basic intuitions (like a symbolic axiomatic logic\(^2\) or a generative grammar) could guide us towards constructing the intended semantic construction of a knowledge representation. So, in KARB, knowledge representation is made up of two ingredients: formal constructions of symbols (lattices of symbols) and generative rules. For example, a piece of knowledge is represented in figure 2.

---

\(^1\) Lattices are mathematical models for constructions in intuitionistic semantics.

\(^2\) Like a rewriting logic
As an underlying philosophy, in KARB’s point of view, the reality and its meanings are composed of statics-related and dynamics-related meanings. Symbolic Constructions capture the statics-related part of knowledge meanings and the generative rules capture the dynamics-related part of knowledge meanings.

Software Patterns are a representation form or media of technical knowledge. Pattern-based solutions for compliance checking have been addressed in the related researches [77]. Compliance Patterns are a kind of knowledge-capturing tool for compliance assessment.

**The KARB solution**

The result of a rule-based benchmarking in KARB is different from other compliance assessment approaches. While other approaches have a result in form of “yes or no”, “correct or incorrect”, etc., in KARB we consider a pool of quantities as results (derived and generated symbols). So the overall state of working memory at the end of each benchmarking process indicates the evaluations of the understudy compliance-case. So a rigorous and reasoned evaluation but with diverse dimensions and values would be achieved.

Each aspect of compliance concerns could be addressed by a separate rule-based benchmark. Each benchmark draws a simple line on the overall picture of compliance assessment scene. A set of multiple, different and diverse benchmarks could make an applied and realistic compliance assessment of a system. This applied and experimental approach to compliance assessment, which relies on rule-based benchmarks, opens a new space for new sorts of innovative, creative and diverse applied methods for compliance assessment.

In figure 1, a brief meta-model of KARB is provided. The assessment of each compliance requirement reified by a compliance benchmarking which in turn consists of some concrete rule-based benchmarks. So each compliance requirement declares meanings and semantics for a compliance benchmarking and compliance benchmarking assesses it.

Figure 2 - An example of a semantic logic rule and the effect of its application on working memory. The rule specifies the existence of attention-policy and excitement when a speaker uses a wonderful beginning for a sentence [0].
Figure 3: as a class diagram\(^3\) of elements, the meta-model of KARB solution is depicted. There are three main dimensions for KARB elements: Compliance related (Red), Benchmarking related (White) and Semantics related (Green) concepts.

A **Compliance Symbol (CSYM)** abstracts a CCON in a similar sense of atom symbols in Lisp, Objects in OO languages (like Java) and JSON fragments in NoSQL DBs: All of them are units for compositional parts. Each **Compliance Concern (CC)** is in association with some **Compliance Requirements\(^4\) (CR)** which capture the notion and attitude of that concern. For example, **Safety** is a compliance concern and in a zoo, we could define this Compliance Requirement in accordance with it: the zoo animals must not be able to harm or threat the visitors (see Example 1).

Some **Compliance Rules (CRUL)** aggregately define the operational realization of a **Compliance Requirement**. Each CRU defines a more rigorous, concrete and special obligation than a CR. In KARB, The rules are considered to be finer than requirements. The overall shape of a requirement is made up of the limiting lines of its constituting parts (= rules). Each **Compliance Rule** has some **Compliance Concepts (CCON)** in its definition. In computational mechanisms of KARB, a **Compliance Symbol (CSYM)** abstracts a CCON. By using a glue of operators (logical, structural, modalities and any needed one), we could construct a formal definition of a CRUL from CSYMs of its CCONs (See Example 1).

---

\(^3\) UML Class Diagram is a popular tool for concept modeling and meta-modeling.

\(^4\) Compliance Requirement is a known concept in Compliance checking literature and frameworks.
Example 1:

| The System Under Compliance | a Zoo |
|-----------------------------|------|
| CC1                         | Safety |
| CR1                         | The zoo animals must not be able to harm or threat the visitors. |
| CRUL1                       | The cage fences must have the proper specification and conditions. |
| CCONs                       | Cage, fence, proper specification, proper condition |
| CSYMs                       | CE, FE, PS, PC |
| Formal Specification of CRUL1 | X IS-A FE(CE) --> O[PS(X)] AND O[PC(X)] |

In KARB, the manner of Formal specification of a CRUL is based on an intuitionistic logic called “semantic logic”. Technically, it could be viewed as an axiomatic system on symbols with Brouwer–Heyting–Kolmogorov Interpretation for semantics [3]. Symbols are considered to be on behalf of basic intuitions (concepts, entities, objects, things, events, values, quantities, qualities, etc.) and the under study under specification system is viewed as a complex construction of basic intuitions.

Formally, it is sufficient to consider semantic logic consists of 1) a set of symbols (behalf of basic intuitions), and 2) a set of rules on them. Each rule describes a symbol generation action: when the left-side symbolic structure of the rule is ready in the working memory, the right-side symbolic structure is generated and pushed to the lattice of symbols in the working memory (See Figure 2).

**Case Study: Software Quality Evaluation**

Since 1990s, there were a variety of approaches for defining measurable quality: the Quality Function Deployment, the Goal Question Metric, and the Software Quality Metrics [81].

Quality Definitions could be seen as a hierarchical formal system of interrelated concepts. This view let us create an explicitly defined conceptual construction for Qualities: a concept-quantized definition for qualities. So a quantification of qualities (which is a well-known but poorly-achieved goal for rigorous software engineering) helps to measure and understand the true level of qualities in each software.

After definition, it is the time for operationalization. Each theoretical concept has its real instances on the ground. User’s feedbacks, comments, experiences, requests, requirements, desires, cognitions and intuitions could help this grounding operationalization. So a good quality-definition theory needs a good quality-grounding theory.

There is a semantic gap between definition theories and grounding theories. The first has a neat nature and the last has a scruffy one[]. How we could bridge the neats and scruffies? A glue model could come forward to resolve this challenge. This model must contain the main conceptual elements of the both sides and try to relate them in a gradient conceptual spectrum. Because this is exactly the manner of KARB solution, we could use it as a method to design a software quality evaluation technique. It is also a kind of evaluation for KARB: it demonstrates its usefulness for a real concern or problem of software engineering community.
**Example 2:** as an example, we provide a semantic logic to model the semantics of this scenario:

Using SMS-Based Dynamic Passwords for E-Banking Transactions. This logic contains rules and intuitions from four context theories: Mobile Apps, Deontic Predicate Logic, Security and The System. The stateless model checking of this scenario semantics (by symbolic-value generations) yields a “false” value. So there is a contradiction in this scenario. Explainable results are provided in the proof construction lattice in fig-4.

1. \( \exists M \in SMS \cdot O(Confidential(M)) \)
2. \( O(X) \rightarrow \neg P(\neg X) \)
3. \( \neg \text{Confidential}(X) \leftrightarrow P(\text{Breach}(X)) \)
4. \( P(P(X)) \rightarrow P(X) \)
5. \( \forall X \in \text{Installed\_Apps}, \forall M \in SMS \cdot P(\text{Access}(X, M)) \)
6. \( (\exists X \in X \cdot \text{Authorized}(L) \land (\text{Access}(X, L))) \rightarrow \text{Breach}(L) \)
7. \( \exists X \in \text{Malwares} \cdot P(X \in \text{Installed\_Apps}) \)
8. \( \forall X \in \text{Malwares}, \forall L \cdot X \notin \text{Authorized}(L) \)
9. \( (A \rightarrow B) \rightarrow (P(A) \rightarrow P(B)) \)
10. \( (\forall X \in K \cdot A(X)) \land (\exists Y \in Q \cdot P(Y \in K)) \rightarrow \exists Y \in Q \cdot P(A(Y)) \)

Although there are some not-mentioned reasoning operations, they are omitted for the sake of simplicity in this preliminary example.


**IR-QUMA Study**

Case study is a popular evaluation method in software engineering research. Case studies are frequently used in papers to demonstrate the capabilities of new techniques and methods [82]. In order to demonstrate and investigate the manner of KARB solution, we conducted a case study. The IR-QUMA study (Iranian Survey on Quality in Messenger Apps) is defined to evaluate the quality of some messenger apps. It consists of these stages:

1. **Selection of some Messenger apps.** The selected apps were Telegram, WhatsApp, Eita, Soroush, Baleh and other popular mobile messenger apps in Iranian Cyberspace. We chose these apps for IR-QUMA case study because we had access to a large community of their users.
2. **Data Gathering:** we design an online questionnaire to collect the opinion of users and as a trace or specification of their user-experience.
3. **Application of KARB solution**
   a. **Elicitation of involving semantic theories**
   b. **Specification of involving semantic theories.** KARB-based specifications was developed for each of involving semantic theories.
      i. KARB-based Specification of Messenger Apps
      ii. KARB-based Specification of some Quality Terms
      iii. KARB-based Specification of User Behavior
      iv. KARB-based Specification of some pieces of HCI knowledge
      v. KARB-based Specification of Risks and Threats
      vi. KARB-based Specification of Software Platform and Mechanisms
      vii. KARB-based Specification of Cognitive Aspects
      viii. KARB-based Specification of Social Aspects
   c. **Computation and Model Checking:** By KARB solution, we computed some of the compliance anomalies.
4. **Evaluation of the results.** We compared the results with three bases: the experts’ judgments, IT reports, and users’ opinions.
IR-QUMA Data Gathering

A questionnaire was designed to evaluate some quality-related measures, metrics and features from users’ experience point of view. The questionnaire was published in popular channels of Iranian mobile social networks, in 10 different messengers (including Telegram, Whatsapp, Instagram, Eita, Soroush, Bale, Gap, IGap, Shaad and Rubika). More than 40 communities of users in these 10 messengers (which are shaping more than 350 micro-communities based on visiting-hour-and-time and spatial partitions) have contributed to this research questionnaire. Total of data is exceeds the level of 7k filled online forms. In our dataset [85], for the sake of data privacy and protection reasons, we hash the name of these messengers by assigning ID-codes from M1 to M10.

A set of statistical analysis, time series analysis, frequency analysis, cluster analysis, classification analysis, geometry locus of data points and topological data analysis are done on these users’ opinion data to obtain useful insights. For example, a correlation locus analysis for two of quality measures for 7 different Messenger apps, in 350 micro-communities, are depicted in fig-5. Each point is in accordance to measure values obtained from one of micro-communities. The blue points are for the one understudy messenger and the red points are for the entire data space (for all understudy messengers). Each axis is demonstrated a 5-level measure value (obtained from averaging of users’ opinions in a micro-community).

In fig-6, we depict this analysis for two other measures: correctness versus quality. It suggests that there is a buffer between “correctness increase\decrease” and “overall quality increase\decrease”. This means that the other factors (than correctness) could play an important role in the overall quality of a software.

![Figure 5- Correlation between Absolut quality measure and relative quality measure, from users’ point of view, for some messenger apps, from IR-QUMA Data.](image-url)
Figure 6- Correlation between Correctness measure and Absolute Quality measure, from users’ point of view, for some messenger apps, from IR-QUMA Data.

Figure 7- Histogram Analysis of quality score levels (obtained from users judgments) about absolute quality measure, for 10 different understudy messenger apps. Each messenger app has 5 data-points for numeration of 1-very poor, 2-poor, 3-moderate, 4-good and 5-excellent quality scores. These scores reflect the users’ experience point of view to the quality of the understudy messenger.
Evaluation of the Method

The KARB solution is applied (based on the semantic logic which is specified in previous steps of study) and the performance of it to correctly calculate (or mimic) the user opinions is evaluated (in terms of error percentage of Benchmark-Computed quality scores and user opinions about quality score). We consider this experimentation setting: 4 different methods for Quality Benchmarking and 5 different experiment-runs (for 5 different messengers).

Each user opinion records involves two sections: 1) user opinion about quality score (which is we named the absolute quality score), and 2) quality context. With quality context we means the factors that could affect or relate to user opinion about quality score: age, gender and other information gathered from the user about her his experience with the messenger (including score of bug-freeness and error-freeness, score of perceived UI complexity, score of rationality of routines, score of usability and ...). The value options for all scores in the questionnaires are defined in a 5-valued Likert scale [83] (Likert scale has been used in various domains of software engineering, for example see [84]). The structure of a user opinion record is provided in figure 8.

![Figure 8- The structure of User Opinion Record](image)

**Definition 1.** \( N_i = \text{number of user opinion records about } App_i \)

**Definition 2.** \( \text{Error\_Percentage}(App_i, Method_j) = \)

\[
\sum_{k=1}^{N_i} \left| \text{Benchmark\_Computed\_Quality}(App_i, Method_j, Context_k) - \text{User\_Quality\_Opinion}_k \right| \]

\( N_i \)

The results of evaluation (by above mentioned experimentation setting) is provided in table-1. These results suggest that the data-driven KARB method could reduce the error percentage, significantly. The error reduction curves (for 5 experiments) are depicted in figure 9. The average-curve for these 5 curves show a pseudo-Sigmoid form. This suggests that the Hybrid Method of DD-KARB (with combination of semantics-awareness and data-drivenness) is more effective than solo-methods and could compute a somehow good estimation for messenger-apps user quality scores. So DD-KARB could be considered as a method for quality benchmarking in this technical context.
Table 1: the results of evaluation (for Experimentation-Plan-ID-1)

| Type of Benchmark          | Error Percentage |
|----------------------------|------------------|
|                            | Exp. 1 | Exp. 2 | Exp. 3 | Exp. 4 | Exp. 5 | Average |
| Quantitative, Based on Stats |       |       |       |       |       |         |
| IR-QUMA (Simple Average)    | 18.4   | 24.3   | 26.3   | 31.8   | 30.1   | 26.2    |
| Qualitative, Based on Sense |       |       |       |       |       |         |
| Expert Quality Scoring      | 17.5   | 23.1   | 25.4   | 30.2   | 29.8   | 25.2    |
| Declarative, Based on Analysis |     |       |       |       |       |         |
| KARB + Hill Climbing        | 11.8   | 15.6   | 15.8   | 17.9   | 16.4   | 15.5    |
| Hybrid, Based on Solving    |       |       |       |       |       |         |
| Data-Driven KARB            | 8.8    | 13.2   | 13.5   | 16.8   | 14.0   | 13.3    |

Figure 9: Error Reduction Curves for 5 different experiments and Average of them.
References

[0] Besharati, M.R., Izadi, M., DAST Paper, 2017.
[1] Sim, Susan Elliott, Steve Easterbrook, and Richard C. Holt. "Using benchmarking to advance research: A challenge to software engineering." In Software Engineering. 2003. Proceedings. 25th International Conference on, pp. 74-83. IEEE, 2003.
[2] Correia, José Pedro, and Joost Visser. "Benchmarking technical quality of software products." In Reverse Engineering, 2008. WCRE'08. 15th Working Conference on, pp. 297-300. IEEE, 2008.
[3] Sato, Masahiko. "Classical Brouwer-Heyting-Kolmogorov interpretation." In Algorithmic Learning Theory, pp. 176-196. Springer Berlin Heidelberg, 1997.
[4] Patterson, David. "For better or worse, benchmarks shape a field: technical perspective." Communications of the ACM 55, no. 7 (2012): 104-104.
[5] Furia, Carlo A., Martin Nordio, Nadia Polikarpova, and Julian Tschannen. "AutoProof: auto-active functional verification of object-oriented programs." International Journal on Software Tools for Technology Transfer 19, no. 6 (2017): 697-716.
[6] Wang, Yangjun. "Stream Processing Systems Benchmark: StreamBench." (2016).
[7] Weide, Bruce, Murali Sitaraman, Heather Harton, Bruce Adcock, Paolo Bucci, Derek Bronish, Wayne Heym, Jason Kirschenbaum, and David Frazier. "Incremental benchmarks for software verification tools and techniques." Verified Software: Theories, Tools, Experiments (2008): 84-98.
[8] Baggen, Robert, José Pedro Correia, Katrin Schill, and Joost Visser. "Standardized code quality benchmarking for improving software maintainability." Software Quality Journal 20, no. 2 (2012): 287-307.
[9] Darmont, Jérôme. "Data-Centric Benchmarking." In Encyclopedia of Information Science and Technology, Fourth Edition, pp. 1772-1782. IGI Global, 2018.
[10] Almeida, Raquel, and Marco Vieira. "Benchmarking the resilience of self-adaptive software systems: perspectives and challenges." In Proceedings of the 6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, pp. 190-195.
[11] Whitman, Michael, and Herbert Mattord. Readings & Cases in Information Security: Law & Ethics. Nelson Education, 2010.
[12] Hashmi, Mustafa, Guido Governorati, Ho-Pun Lam, and Moe Thandar Wynn. "Are We Done with Business Process Compliance: State-of-the-Art and Challenges Ahead." Knowledge and Information Systems (2017).
[13] Yip, Frederick, Alfred Ka Yiu Wong, Nandan Parameswaran, and Pradeep Ray. "Towards Robust and Adaptive Semantic-Based Compliance Auditing," In EDOC Conference Workshop, 2007. EDOC'07. Eleventh International IEEE, pp. 181-188. IEEE, 2007.
[14] Kanoun, Karama, and Lisa Spinthower. Dependability benchmarking for computer systems. Vol. 72. John Wiley & Sons, 2008.
[15] Antunes, Nuno, and Marco Vieira. "Assessing and comparing vulnerability detection tools for web services: Benchmarking approach and examples." IEEE Transactions on Services Computing 8, no. 2 (2015): 269-283.
[16] Friginal, Jesús, David De Andres, Juan-Carlos Ruiz, and Pedro Gil. "Towards benchmarking routing protocols in wireless mesh networks." Ad Hoc Networks 9, no. 8 (2011): 1374-1388.
[17] Nwulu, Equi Emmanuel. "Utility of the HPT Framework for Improving Distance Education in Nigeria." Ph.D. diss., Walden University, 2018.
[18] Johannessen, Marius Rohde, and Lasse Berntzen. "The Transparent Smart City." In Smart Technologies for Smart Governments, pp. 67-94. Springer, Cham, 2018.
[19] Le Goues, Claire, Neal Holtschulte, Edward K. Smith, Yurii Brun, Premkumar Devanbu, Stephanie Forrest, and Westley Weimer. "The ManyBugs and IntroClass benchmarks for automated repair of C programs." IEEE Transactions on Software Engineering 41, no. 12 (2015): 1236-1256.
[20] Price, Walter J. "A benchmark tutorial." IEEE micro 9, no. 5 (1989): 28-43.
[21] Jacobvitz, Adam N., Andrew D. Hilton, and Daniel J. Sorin. "Multi-program benchmark definition." In Performance Analysis of Systems and Software (ISPASS), 2015 IEEE International Symposium on, pp. 72-82. IEEE, 2015.
[22] Tichy, Walter F. "Should computer scientists experiment more?." Computer 31, no. 5 (1998): 32-40.
Zeuch, Matthias, ed. Handbook of human resources management. Springer, 2016.

Prasad, T. V. V. V., and Raghu B. Korrapati. "Computerized Applications of Natural Language Processing in Digital Economy: A Review." International Journal of Engineering and Management Research (IJEMR) 7, no. 1 (2017): 239-241.

Nash, Edward, Jens Wiebensohn, Raimo Nikkiä, Anna Vatsanidou, Spyros Fountas, and Ralf Bill. "Towards automated compliance checking based on a formal representation of agricultural production standards." Computers and electronics in agriculture 78, no. 1

Maxwell, Jeremy C., and Annie I. Anton. "Developing production rule models to aid in acquiring requirements from legal texts." In Requirements Engineering Conference, 2009. RE'09. 17th IEEE International, pp. 101-110. IEEE, 2009.

Ingolfo, Silvia. "Nomos 3: legal compliance of software requirements." Ph.D. diss., University of Trento, 2015.

Ingolfo, Silvia, Alberto Siena, Angelo Susi, Anna Perini, and John Mylopoulos. "Modeling laws with nomos 2." In Requirements Engineering and Law (RELAW), 2013 Sixth International Workshop on, pp. 69-71. IEEE, 2013.

Siena, Alberto, Anna Perini, Angelo Susi, and John Mylopoulos. "A meta-model for modelling law-compliant requirements." In Requirements Engineering and Law (RELAW), 2009 Second International Workshop on, pp. 45-51. IEEE, 2009.

Stratigaki, Christini, Pericles Loucopoulos, and Maria Nikolaidou. "Designing a Meta Model as the Foundation for Compliance Capability." In 1st International Workshop on Capability-oriented Business Informatics organized in conjunction with the 16th IEEE

Papazoglou, Michael P. "Making business processes compliant to standards and regulations." In Enterprise Distributed Object Computing Conference (EDOC), 2011 15th IEEE International, pp. 3-13. IEEE, 2011.

Turetken, Oktay, Amal Elgammal, Willem-Jan van den Heuvel, and Michael P. Papazoglou. "Capturing compliance requirements: A pattern-based approach." IEEE Software 29, no. 3 (2012): 28-36.

Fowler, Martin. Domain-specific languages. Pearson Education, 2010.

Eastman, Chuck, Jae-min Lee, Yeon-suk Jeong, and Jin-kook Lee. "Automatic rule-based checking of building designs." Automation in construction 18, no. 8 (2009): 1011-1033.

Ismail, Aimi Sara, Kherun Nita Ali, and Noorminshah A. Iahad. "A Review on BIM-based automated code compliance checking system." In Research and Innovation in Information Systems (ICRIIS), 2017 International Conference on, pp. 1-6. IEEE, 2017.

Zhioua, Zeineb, Yves Roudier, Rabéa Amour-Boulifa, Takoua Kechiche, and Stuart Short. "Tracking Dependent Information Flows." In ICISSS, pp. 179-189. 2017.

Arnett, Kirk P., Gary F. Templeton, and David A. Vance. "Information security by words alone: The case for strong security policies." Security and Privacy Assurance in Advancing Technologies: New Developments: New Developments (2010): 154.

Zhang, Jiansong, and Nora M. El-Gohary. "Semantic-based logic representation and reasoning for automated regulatory compliance checking." Journal of Computing in Civil Engineering 31, no. 1 (2016): 04016037.

Preidel, Cornelius, and André Bormann. "Refinement of the Visual Code Checking Language for an Automated Checking of Building Information Models Regarding Applicable Regulations." In Computing in Civil Engineering 2017, pp. 157-165. 2017.

Kokash, Natallia, and Farhad Arbab. "Formal behavioral modeling and compliance analysis for service-oriented systems." In International Symposium on Formal Methods for Components and Objects, pp. 21-41. Springer, Berlin, Heidelberg, 2008.

Zhang, Jiansong, and Nora M. El-Gohary. "Integrating semantic NLP and logic reasoning into a unified system for fully-automated code checking." Automation in Construction 73 (2017): 45-57.

De Silva, Lakshitha R. "Towards controlling software architecture erosion through runtime conformance monitoring." Ph.D. diss., University of St Andrews, 2014.

Dardar, Raghad. "Building a safety case in compliance with iso 26262 for fuel level estimation and display system." (2014).

Kelly, Timothy Patrick. "Arguing safety: a systematic approach to managing safety cases." Ph.D. diss., University of York, 1999.
[76] Kokaly, Sahar, Rick Salay, Mehrdad Sabetzadeh, Marsha Chechik, and Tom Maibaum. "Model management for regulatory compliance: a position paper." In Proceedings of the 8th International Workshop on Modeling in Software Engineering, pp. 74-80. ACM, 2016.

[77] Elgammal, Amal, Oktay Turetken, and Willem-Jan Van Den Heuvel. "Using patterns for the analysis and resolution of compliance violations." International Journal of Cooperative Information Systems 21, no. 01 (2012): 31-54.

[78] Giblin, Christopher, Alice Y. Liu, Samuel Müller, Birgit Pfitzmann, and Xin Zhou. "Regulations Expressed As Logical Models (REALM)." In Jurix, pp. 37-48. 2005.

[79] Nawaz, Falak, Ahmad Mohsin, Syda Fatima, and Naeem Khalid Janjua. "Rule-based multi-criteria framework for SaaS application architecture selection." In IFIP International Conference on Artificial Intelligence in Theory and Practice, pp. 129-138. Springer, Cham, 2015.

[80] Ingolfo, Silvia, Alberto Siena, Angelo Susi, Anna Perini, and John Mylopoulos. "Modeling laws with nomos 2." In 2013 6th International Workshop on Requirements Engineering and Law (RELAW), pp. 69-71. IEEE, 2013.

[81] Caldiera, Victor R. Basili Gianluigi, and H. Dieter Rombach. "The goal question metric approach." Encyclopedia of software engineering (1994): 528-532.

[82] Perry, Dewayne E., Susan Elliott Sim, and Steve M. Easterbrook. "Case studies for software engineers." In Proceedings. 28th International Conference on Software Engineering (ICSE’06), pp. 1045-1046. 2006.

[83] Likert, Rensis. "A technique for the measurement of attitudes." Archives of psychology (1932).

[84] Gren, Lucas, Richard Torkar, and Robert Feldt. "The prospects of a quantitative measurement of agility: A validation study on an agile maturity model." Journal of Systems and Software 107 (2015): 38-49.

[85] IR-QUMA Data-Set. DiSysLab, Sharif University of Technology, 2020.

[86] Nowroozi, Alireza, Peyman Teymoori, Toktam Ramezanifarkhani, Mohammad Reza Besharati, and Mohammad Izadi. "A Crisis Situations Decision-Making Systems Software Development Process With Rescue Experiences." IEEE Access 8 (2020): 59599-59617.