End-to-End Regulatory Compliance Evolution Pipeline: From Legal Requirements to Developer Tests

Carolin Brandt
c.e.brandt@tudelft.nl
Delft University of Technology
The Netherlands

Sallam Abualhaija
sallam.abualhaija@uni.lu
University of Luxembourg
Luxembourg

Abstract

As artificial intelligence agents become more prevalent, software systems are becoming even more deeply integrated in our lives while also growing in complexity. In response to societal concerns about safety and privacy, new regulations are being regularly introduced, posing significant compliance challenges for software engineers. This challenge is multifaceted, spanning the interpretation of legal texts, determination of how regulations apply to a specific software product, validation of end-to-end compliance, and, at the lowest level, the need to guide developers throughout this process. This challenge is exacerbated by the constant changes in regulations, requirements, and software systems. We posit that an end-to-end integration of requirements engineering and software testing, down to developer practices, could advance regulatory compliance for software systems. In this vision paper, we present a workflow that operationalizes constantly evolving regulatory requirements into automated developer-focused tests. The novelty of our approach lies in the orchestration of requirements elicitation, regulatory interpretation, traceability, test generation, and developer feedback into a continuous compliance workflow.

CCS Concepts

• **Social and professional topics** → **Governmental regulations**; • **Software and its engineering** → **Requirements analysis**; **Software testing and debugging**; **Software evolution**.

Keywords

Regulatory Compliance, The AI Act, GDPR, Test Case Generation, Software Testing, Requirements Engineering, Software Evolution

ACM Reference Format:
Carolin Brandt and Sallam Abualhaija. 2026. End-to-End Regulatory Compliance Evolution Pipeline: From Legal Requirements to Developer Tests. In *Proceedings of the 41st IEEE/ACM International Conference on Automated Software Engineering (ASE '26), October 12–16, 2026, Munich, Germany*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3832783.3834541

1 Introduction

As artificial intelligence (AI)-enabled software becomes integral to our lives, ethical concerns around privacy, safety, and manipulation, have become prominent. In a recent survey, 84% of participants reported being concerned about their privacy, and many expressed worries about transparency (37%) and manipulation (42%) [30]. Other surveys reported similar findings [9, 46]. Many of these concerns are explicitly addressed and, in parts regulated, by the General Data Protection Regulation (GDPR) [63] and the Artificial Intelligence Act (AIA) [64], two predominant European regulations with far-reaching global implications. Current software applications provide personalized services and smart assistance across domains, ranging from safety-critical to non-critical, and have thus become key targets for compliance with these and other regulations.

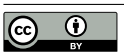
Regulatory compliance has been widely studied in the software engineering (SE) literature, often through isolated efforts split between upstream and downstream stages of software development.

On the *requirements engineering (RE)* side, existing work addresses the elicitation of legal requirements [3, 65, 66], their representation [19, 36, 56, 58, 72], and, to some extent, their operationalization [34, 47]. Much of this work remains at the level of generic legal requirements or compliance checklists, leaving implementation details to the developers' interpretation and creating thereby potential risks of non-compliance [3]. The literature also acknowledges the challenging yet inevitable need to involve legal experts in interpreting and contextualizing regulation into implementable and verifiable compliance requirements [20]. This need is particularly emphasized when software systems must comply with multiple evolving regulations that may overlap or conflict, a complexity that often occurs but is oversimplified in the existing literature.

On the *software testing* side, existing work addresses compliance verification in software [73], e.g., leveraging formal methods [32, 39], or code analysis [61]. However, this research direction often lacks involvement from RE and overlooks the important role of developers throughout the process. Unit tests could facilitate communication about compliant behavior of components. However, existing requirements-based test generation approaches mainly produce abstract test cases that need to be manually executed [69, 71]. The tests remain at the system level, and approaches to create and maintain compliance unit tests [62], meaningful for developers, remain lacking. A challenge lies in the scale of modern systems where many components and teams contribute to compliance [67].

The fragmented nature of these efforts has left persistent gaps between legal requirements derived from regulations, on the one hand, and what is actually implemented in software systems, on the other [27, 43]. These gaps are evident in practice. For instance, a recent survey reports GDPR compliance violations in how mobile applications privacy policies disclose individual rights, which may in turn mirror implementation-level violations [21]. Similar implementation challenges have been documented for the AIA [45, 54, 68].

In this vision paper, we propose a workflow that brings together law, RE, and developer-centered software testing. We call it *RELAI*,



ASE '26, Munich, Germany

ACM ISBN 979-8-4007-2882-2/2026/10
https://doi.org/10.1145/3832783.3834541

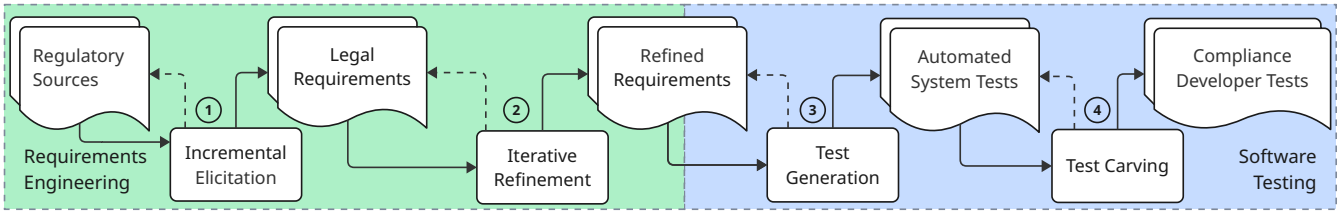


Figure 1: Overview of the end-to-end software compliance workflow.

which stands for *tRanslating legal rEquirements to deveLoper-level test units using AI*. RELAI provides automated support [8, 49] for building compliant-by-design AI-enabled systems and for continuously verifying compliance of systems in response to regulatory changes [2] or code changes [50]. To this end, we propose leveraging *AI agents* to reduce the overhead of manual validation and improve efficiency in the compliance process, while humans-in-the-loop preserve oversight and responsible decision making.

Specifically, our proposed workflow RELAI realizes an end-to-end software compliance workflow to operationalize legal requirements on one end, verify compliance through software tests on the other, and support both evolving regulations and software artifacts, as illustrated in Figure 1. RELAI is composed of four steps: ① Regulation is interpreted, contextualized, and translated into software requirements. ② These requirements are refined and formalized with respect to a particular software, company, and domain. ③ Based on these requirements, matching high-level automated tests for the entire system are generated. ④ These tests are then executed and carved into more focused unit tests aligned with the developers' contexts. This process incorporates feedback loops to update artifacts based on the impact of software or regulation evolution, informed by trace links between all artifacts.

The novelty of RELAI lies in the orchestration of the above activities. Its contribution to existing research on regulatory compliance is two-fold: First, we introduce *Addressee-focused*[1] requirements elicitation, aligning RE with legal practices. Second, we propose *test carving* as a means to generate developer-centric unit tests from high-level compliance tests [26]. Next, we position our proposed ideas against existing literature and present the workflow in detail.

2 Related Work

This section provides a snapshot of literature on regulatory compliance to motivate and ground our proposed workflow.

2.1 Operationalizing Regulations for Software

Existing work investigates the extraction of compliance-related information and requirements from legal text [5, 12, 58], supported by formalized representations of legal provisions [17, 52, 57]. Most of these approaches largely focus on privacy regulations like the Health Insurance Portability and Accountability Act (HIPAA) and GDPR, while AIA has recently attracted attention through studies interpreting its relevance for SE (e.g., [45, 54, 68]). The advances in AI enabled new directions like privacy requirements generation [53], regulatory changes analysis [2], and the transformation of legal text into code representations [56] or other formal language [69]. While the concept of *addressee* has emphasized in this context, its explicit integration into the contextualization of legal requirements remains underexplored (*research gap 1*), creating a gap between regulatory analysis in the literature and compliance implementation in practice [20, 21].

The operationalization and refinement of legal requirements has also been studied. Prior work aims to better align system requirements with legal requirements through goal modeling [4], as well as through retrieval-augmented generation with curated legal sources to extract a comprehensive set of legal requirements [3]. Closely related efforts address the formalization of regulatory requirements [42], and the integration of human-values into software development [10]. While existing work emphasizes the need to collaborate with legal experts [18, 20], enabling this collaboration remains underexplored (*research gap 2*). In particular, automated support, shared intermediary languages, and the use of additional authoritative legal sources to increase confidence in building compliant software remain insufficiently addressed.

2.2 Developer-centric Compliance Testing

Existing compliance processes primarily use modeling, stakeholder involvement [40], conflict detection [31], or query-analysis of knowledge graphs [51], which all need human involvement. Literature calls for automating compliance management to reduce costs, avoid errors, and address challenges in continuous delivery and multi-environment management [8, 49, 60]. However, generating executable tests from requirements remains an open challenge (*research gap 3*) [71]. Especially privacy and AI regulations receive less attention in automated compliance compared to security [6, 50, 60].

The typical focus of requirements-based test generation on system tests [69, 71] and compliance checking at the release stage [41], lead to a key compliance challenge: Lack of awareness among developers [1, 67]. To our knowledge, literature has not yet explored in how far established approaches to generate low-level tests, like test carving [26, 28, 70], can address compliance testing and awareness among developers (*research gap 4*). Test carving creates unit and API-level tests that match system tests, with similar coverage, faster execution [70] and meaningful test data [26]. When generating developer-centric tests, a crucial aspect is to select tests that cover relevant behavior [13–15], for which visualizations and trace execution data are promising [16, 37], but not yet investigated.

3 End-to-end Software Compliance Vision

This section introduces our end-to-end software compliance workflow RELAI. It takes regulatory sources (e.g., laws, regulations, guidelines and standards) as input, which over four steps are transformed into developer tests that capture compliance-related behavior of software components. They are connected to corresponding high-level compliance tests and contextualized, refined regulatory

[1] *Addressee* defines the person(s) to which a document is addressed.

requirements. These links enable (i) traceability between regulations and their operationalization in software implementations, enabling efficient compliance audits, (ii) automated, multi-layered compliance checks suited for continuous compliance checking of changing software systems, and (iii) operationalizations of regulations that are highly developer-centric through being encoded as executable verification of a software component.

3.1 Operationalizing Legal Requirements for Software Compliance

Translating regulation into software requirements requires shared understanding between legal experts and requirements engineers. Our workflow supports automated end-to-end compliance in two steps: (i) eliciting contextualized legal requirements tailored to a specific software application, and (ii) refining high-level requirements into concrete software requirements grounded in legal sources.

3.1.1 Eliciting Contextualized Legal Requirements. This step derives a complete set of legal requirements relevant to a specific organization, such as *providers* or *deployers* of AI systems [68]. We focus only on requirements that can be implemented in software. Since this process requires knowledge of both the system and the applicable regulations, it should be carried out jointly by requirements engineers, who validates the context, and legal experts, who validate the legal interpretation and resulting requirements. The involvement of legal experts can be facilitated through established interdisciplinary collaborations, for which lessons learned have been documented in prior work (see [55]).

Current AI technologies have demonstrated promising capabilities in understanding text, including legal statements [44]. To establish shared knowledge among different experts, a possible research direction is to build AI agents for **creating intermediate representations**. This could involve an agent for **rewriting legal texts with respect to a specific addressee**, on the one hand, and another for rewriting technical documentation about the software system, on the other, to capture relevant roles, responsibilities, and actions. In this way, it becomes explicit *who* must do *what*, *under which circumstances*, and *with which evidence*.

Key challenges include: (i) Regulatory obligations are often interconnected and distributed across multiple provisions, including different articles within the same regulation and references to external regulations. We envision **extract obligations together with their context of applicability**, such as conditions and exceptions. (ii) Regulation changes: new provisions, amendments, and guidelines may alter the applicability or content of previously derived obligations. We therefore aim to build **AI-enabled incremental elicitation methods** that identify regulatory changes, analyze their impact on elicited legal requirements, and update only the affected ones.

> *RQ1. How can incrementally derive comprehensive, addressee-centric, machine-analyzable legal requirements from evolving regulations, informed by software technical documentation?*

We propose evaluating this RQ by comparing between the automatically extracted addressee-centric requirements and manually-elicited ones in terms of completeness and software-relevance.

3.1.2 Refining Legally-grounded Software Requirements. This step iteratively refines the extracted legal obligations into implementable software requirements that can be systematically integrated into subsequent software engineering activities. While software requirements are concrete and context-dependent, regulations typically state generic obligations and leave key implementation details underspecified. Moreover, both the regulations and software systems may evolve. Refinement is therefore an iterative activity that connects regulatory changes with changes in software artifacts.

We foresee two research opportunities. The first is to **leverage additional authoritative sources**, such as guidelines, technical standards, and relevant case law, to better understand these obligations for a specific software system and its context [3]. The second is to **complement top-down legal interpretation with bottom-up evidence from the state of practice**.

Key challenges include: (i) Regulations are often subject to interpretation, depending on the specific application context. We therefore propose **defining refined requirements with varying degrees of compliance** [59], corresponding to the legal sources from which the requirements are extracted. (ii) Compliance state of practice may vary across contexts, resulting in various implementations emphasizing the interpretative variability described in (i). We envision **pursuing empirical methods**, including interviews and surveys, to better understand existing compliance practices (iii) Regulations and software artifacts change independently. We therefore aim to support **bidirectional change impact analysis**, where regulatory changes are traced to the software requirements and artifacts they affect, and software changes are traced back to the legal obligations whose interpretation or implementation may need to be revised.

> *RQ2. How can we systematically and iteratively refine legal requirements derived from multiple regulations within a concrete and evolving software application context?*

This RQ can be evaluated through coverage assessments, comparing automatically-derived detailed requirements against existing ones implemented in a given software system.

3.2 Generating Developer Tests for Compliance

Based on contextualized software requirements, our workflow supports compliance testing in two steps: (i) automatically translate legal requirements into high-level system tests, validated by domain experts, and (ii) carve these into low-level tests that provide fast, actionable feedback to developers on component-level compliance.

3.2.1 Defining System Tests for Regulatory Compliance. This step transforms regulatory requirements into automated high-level tests that exercise the entire system through the user interface and validates them with domain experts. The validation process and traceability to requirements play a key role in creating a shared understanding of the regulations between legal experts and engineers and trust in the automated compliance process.

Generative AI can support the generation of automated tests from requirements. However, it remains prone to hallucinations and other issues requiring substantial manual intervention from engineers [38], and can increase the burden on experts to validate their outputs [35], which may outweigh the value of automatic

generation [13]. Research opportunities include exploring and validating **visualization techniques for lightweight test validation by experts**, leveraging automated test recordings and automatic transfer of validations between similar tests.

The challenge in testing is to choose the right concrete examples to be confident the software behaves correctly [7]. For regulatory compliance one execution might not be enough, e.g., to show that each datapoint collected is also used (data minimization from GDPR). We propose learning from current compliance approaches how they select the inputs they exercise to verify compliance. With this, we develop a **compliance-focused understanding of test adequacy** and input space coverage [22, 33].

Automated compliance tests enable continuous compliance checking of evolving software systems, providing feedback to engineers whether changes affect compliance behavior. If regulations change, this step can be re-executed in a diff-focused manner: Leveraging the knowledge from previously validated tests [15, 24] and proposing updates of existing tests, lowering the validation burden [25].

RQ3. How can we generate adequate and automated end-to-end compliance test suites and efficiently validate them with legal experts and engineers?

This RQ can be evaluated by comparing the coverage of generated tests with existing manual compliance checking approaches, and by user studies with engineers and legal experts.

3.2.2 Carving Tests for Developer-centric Continuous and Automated Compliance. To give continuous and meaningful feedback to software developers, the second step carves the end-to-end tests into smaller tests, typically called unit or developer tests [62]. *Test carving* [26, 28, 70] refers to tracing the execution of a high-level test through a software system and defining shorter tests that replay one step of that execution. These unit tests are closely connected to the components that developers work on, making them cheaper to execute. They provide feedback and can help developers understand how compliance concretely impacts their component. This defines a compliance process for sub-systems, and increases awareness among developers about compliance requirements, addressing challenges in software compliance identified by Usman et al. [67]. The connection between the carved test cases, their end-to-end tests, and their regulatory requirements also provide traceability [17, 23, 37] down to the code level, indicating where in a software system compliance is realized. We envision designing and validating code-centric **visualizations** of carved developer tests **that facilitate comprehension of compliance-contribution** [16].

A challenge is to distinguish which behavior is relevant for complying with the regulatory requirement and worth capturing as a regression test [13, 15]. Future studies should explore how to **identify regulation-related behavior** through tagging and data-flow analysis, including reducing engineer input **by leveraging semantic information** in code and documentation. This information can also be used for test selection and prioritization [11, 29, 48].

This automated step allows developers to efficiently update their tests to changes in regulations. Carving failing compliance system tests, e.g., after a regulation change, helps developers reason about how to address the change in their component.

RQ4. How can we carve effective developer tests for compliance, that capture relevant behavior and inform developers about the compliance contribution of their components?

This can be evaluated by measuring the developers' effectiveness of capturing and localizing compliance regressions with carved tests compared to high-level tests.

4 Application Scenario

To illustrate the complex scenarios that we envision RELAI to contribute to, consider an AI companion for seniors which offers personalized assistance through voice guidance, customized workouts, and motion analysis[2]. AI companions may influence behavior (AIA, high-risk) and process sensitive personal data (GDPR). They are complex software systems composed of multiple interconnected components, ranging from user interfaces to thousands of lines of code. RELAI supports compliance checking by decomposing end-to-end obligations (e.g., transparency toward users) into executable tests that capture how individual components, such as data processing or model inference, contribute to compliance, providing developers with concrete, actionable feedback.

For instance, processing sensitive data may be important to enable the personalized and effective functioning of an AI companion. However, such processing must remain limited under GDPR data minimization principles (e.g., Article 5(1)(c)). This constraint may affect the system's ability to fulfill transparency requirements (AIA, Art. 50). Since the AIA imposes different obligations on providers versus deployers of an AI system, an addressee-centric elicitation can inform compliance testing in this case. For instance, compliance testing for a provider might focus more on the design level, e.g., AI-relevant generated content, audio, text, video, or text, can be technically marked and detectable where required. Testing for a deployer, on the other hand, might focus more on system-user interactions, e.g., checking whether the deployed interface actually informs the user at the right moment or that sensitive data in is exposed only when necessary. The same technical behavior might therefore require different addressee-dependent tests, where provider needs to show that the system is capable of delivering transparent AI-interaction, whereas the deployer needs to show that such a capability is actually used appropriately.

Building compliant AI companions is further complicated by the dynamic nature of both the regulatory and technical landscape: regulations evolve over time, new guidelines and standards emerge, and AI companions themselves continuously evolve, introducing new services and updates. By explicitly linking legal requirements to software tests, RELAI supports targeted updates and continuous reassessment as both regulations and software evolve.

RELAI builds on well-established practices in RE and software testing, providing an end-to-end workflow that *orchestrates* established activities and innovative approaches to support continuous compliance in view of regulatory and software evolution.

[2] NYTimes on ELliQ: ai-robot-senior-companion

Acknowledgments

This research was funded in part by the Luxembourg National Research Fund (FNR), under grants numbers NCER22/IS/16570468/NCER-FT and C23/IS/17958091/PLAITO.

Data Availability Statement

There is no data available yet.

References

[1] Norris Syed Abdullah, Shazia Sadiq, and Marta Indulska. 2010. Emerging Challenges in Information Systems Research for Regulatory Compliance Management. In *Advanced Information Systems Engineering, 22nd International Conference, CAiSE 2010 (LNCS)*. Springer, 251–265. doi:10.1007/978-3-642-13094-6_21

[2] Sallam Abualhaija, Marcello Ceci, Nicolas Sannier, Domenico Bianculli, Lionel C. Briand, Dirk A. Zetzsche, and Marco Bodellini. 2024. AI-Enabled Regulatory Change Analysis of Legal Requirements. In *32nd IEEE International Requirements Engineering Conference, RE 2024*. IEEE, 5–17. doi:10.1109/RE59067.2024.00012

[3] Sallam Abualhaija, Marcello Ceci, Nicolas Sannier, Domenico Bianculli, Salomé Lannier, Martina Siclari, Olivier Voordeckers, and Stanislaw Tosza. 2025. LLM-assisted Extraction of Regulatory Requirements: A Case Study on the GDPR. In *33rd IEEE International Requirements Engineering Conference, RE 2025*. IEEE, 142–154. doi:10.1109/RE63999.2025.00023

[4] Okhaide Akhigbe, Daniel Amyot, and Gregory Richards. 2019. A systematic literature mapping of goal and non-goal modelling methods for legal and regulatory compliance. *Requir. Eng.* 24, 4 (2019), 459–481. doi:10.1007/S00766-018-0294-1

[5] Orlando Amaral, Sallam Abualhaija, Damiano Torre, Mehrdad Sabetzadeh, and Lionel C. Briand. 2022. AI-Enabled Automation for Completeness Checking of Privacy Policies. *IEEE Trans. Software Eng.* 48, 11 (2022), 4647–4674. doi:10.1109/TSE.2021.3124332

[6] Florian Angermeir, Jannik Fischbach, Fabiola Moyón, and Daniel Méndez. 2024. Towards Automated Continuous Security Compliance. In *18th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM 2024*, Xavier Franch, Maya Daneva, Silverio Martínez-Fernández, and Luigi Quaranta (Eds.). ACM, 440–446. doi:10.1145/3674805.3690748

[7] Maurício Aniche. 2022. *Effective Software Testing: A Developer's Guide*. Simon and Schuster.

[8] Julieth Patricia Castellanos Ardila, Barbara Gallina, and Faiz Ul Muram. 2022. Compliance checking of software processes: A systematic literature review. *J. Softw. Evol. Process.* 34, 5 (2022). doi:10.1002/SMR.2440

[9] B Auxier, M Anderson, A Perrin, and E Turner. 2020. Parenting approaches and concerns related to digital devices. *Pew Research Center* (2020).

[10] Amel Bennaceur, Diane Hassett, Bashar Nuseibeh, and Andrea Zisman. 2023. Values@Runtime: An Adaptive Framework for Operationalising Values. In *45th IEEE/ACM International Conference on Software Engineering: Software Engineering in Society, SEIS@ICSE 2023*. IEEE, 175–179. doi:10.1109/ICSE-SEIS58686.2023.00024

[11] Antonia Bertolino, Antonio Guerriero, Breno Miranda, Roberto Pietrantuono, and Stefano Russo. 2020. Learning-to-rank vs ranking-to-learn: strategies for regression testing in continuous integration. In *ICSE '20: 42nd International Conference on Software Engineering*. ACM, 1–12. doi:10.1145/3377811.3380369

[12] Jaspreet Bhatia, Morgan C. Evans, and Travis D. Breaux. 2019. Identifying incompleteness in privacy policy goals using semantic frames. *Requir. Eng.* 24, 3 (2019), 291–313. doi:10.1007/S00766-019-00315-Y

[13] Carolin E. Brandt, Marco Castelluccio, Christian Holler, Jason Kratzer, Andy Zaidman, and Alberto Bacchelli. 2024. Mind the Gap: What Working With Developers on Fuzz Tests Taught Us About Coverage Gaps. In *46th International Conference on Software Engineering: Software Engineering in Practice, ICSE-SEIP 2024*. ACM, 157–167. doi:10.1145/3639477.3639721

[14] Carolin E. Brandt, Ali Khatami, Mairieli Wessel, and Andy Zaidman. 2024. Shaken, Not Stirred: How Developers Like Their Amplified Tests. *IEEE Trans. Software Eng.* 50, 5 (2024), 1264–1280. doi:10.1109/TSE.2024.3381015

[15] Carolin E. Brandt and Andy Zaidman. 2022. Developer-centric test amplification. *Empir. Softw. Eng.* 27, 4 (2022), 96. doi:10.1007/S10664-021-10094-2

[16] Carolin E. Brandt and Andy Zaidman. 2022. How Does This New Developer Test Fit In? A Visualization to Understand Amplified Test Cases. In *Working Conference on Software Visualization, VISSOFT 2022*. IEEE, 17–28. doi:10.1109/VISSOFT55257.2022.00011

[17] Travis D. Breaux and David G. Gordon. 2013. Regulatory Requirements Traceability and Analysis Using Semi-formal Specifications. In *19th International Working Conference on Requirements Engineering: Foundation for Software Quality, REFSQ 2013. (LNCS, Vol. 7830)*. Springer, 141–157. doi:10.1007/978-3-642-37422-7_11

[18] Travis D. Breaux and Thomas B. Norton. 2022. Legal Accountability as Software Quality: A U.S. Data Processing Perspective. In *30th IEEE International Requirements Engineering Conference, RE 2022*. IEEE, 101–113. doi:10.1109/RE54965.2022.00016

[19] Travis D. Breaux, Matthew W. Vail, and Annie I. Antón. 2006. Towards Regulatory Compliance: Extracting Rights and Obligations to Align Requirements with Regulations. In *Proceedings of RE 2006*. IEEE, 46–55. doi:10.1109/RE.2006.68

[20] Marcello Ceci, Nicolas Sannier, Sallam Abualhaija, Domenico Bianculli, and Michael Halling. 2025. When Data Stands Before the Law: An Experience Report on Representing Financial Rules in SPARQL. In *Companion Proceedings of the 9th International Joint Conference on Rules and Reasoning, RuleML+RR 2025 (CEUR Workshop Proceedings, Vol. 4083)*. CEUR-WS.org. https://ceur-ws.org/Vol-4083/paper66.pdf

[21] Orlando Amaral Cejas, Sallam Abualhaija, Nicolas Sannier, Marcello Ceci, and Domenico Bianculli. 2025. GDPR Compliance in Privacy Policies of Mobile Apps: An Overview of the State-of-Practice. In *33rd IEEE International Requirements Engineering Conference, RE 2025*. IEEE, 320–331. doi:10.1109/RE63999.2025.00038

[22] Tsong Yueh Chen, Fei-Ching Kuo, Robert G. Merkel, and T. H. Tse. 2010. Adaptive Random Testing: The ART of test case diversity. *J. Syst. Softw.* 83, 1 (2010), 60–66. doi:10.1016/J.JSS.2009.02.022

[23] Jane Cleland-Huang, Adam Czauderna, Marek Gibiec, and John Emenecker. 2010. A machine learning approach for tracing regulatory codes to product specific requirements. In *32nd ACM/IEEE International Conference on Software Engineering, ICSE 2010*. ACM, 155–164. doi:10.1145/1806799.1806825

[24] Benjamin Danglot, Oscar Vera-Perez, Zhongxing Yu, Andy Zaidman, Martin Monperrus, and Benoit Baudry. 2019. A snowballing literature study on test amplification. *J. Syst. Softw.* 157 (2019). doi:10.1016/J.JSS.2019.110398

[25] Benjamin Danglot, Oscar Luis Vera-Pérez, Benoit Baudry, and Martin Monperrus. 2019. Automatic test improvement with DSpot: a study with ten mature open-source projects. *Empir. Softw. Eng.* 24, 4 (2019), 2603–2635. doi:10.1007/S10664-019-09692-Y

[26] Amirhossein Deljouyi and Andy Zaidman. 2023. Generating Understandable Unit Tests through End-to-End Test Scenario Carving. In *23rd IEEE International Working Conference on Source Code Analysis and Manipulation, SCAM 2023, Bogotá*. IEEE, 107–118. doi:10.1109/SCAM59687.2023.00021

[27] Konstantin Dmitriev, Shanza Ali Zafar, Kevin Schmiechen, Yi Lai, Micheal Saleab, Pranav Nagarajan, Daniel Dollinger, Markus Hochstrasser, Stephan Myschik, and Florian Holzapfel. 2020. A Lean and Highly-automated Model-Based Software Development Process Based on DO-178C/DO-331. *CoRR* abs/2010.06505. arXiv:2010.06505 https://arxiv.org/abs/2010.06505

[28] Sebastian G. Elbaum, Hui Nee Chin, Matthew B. Dwyer, and Matthew Jorde. 2009. Carving and Replaying Differential Unit Test Cases from System Test Cases. *IEEE Trans. Software Eng.* 35, 1 (2009), 29–45. doi:10.1109/TSE.2008.103

[29] Sebastian G. Elbaum, Alexey G. Malishevsky, and Gregg Rothermel. 2002. Test Case Prioritization: A Family of Empirical Studies. *IEEE Trans. Software Eng.* 28, 2 (2002), 159–182. doi:10.1109/32.988497

[30] Yagil Elias, Tom P. Humbert, Lauren Olson, and Emitzá Guzmán. 2025. What Is Unethical about Software? User Perceptions in the Netherlands. In *IEEE/ACM International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS '25)*. IEEE Press, 118–129. doi:10.1109/ICSE-SEIS66351.2025.00017

[31] Priscila Engiel, Julio César Sampaio do Prado Leite, and John Mylopoulos. 2017. A tool-supported compliance process for software systems. In *11th International Conference on Research Challenges in Information Science, RCIS 2017*. IEEE, 66–76. doi:10.1109/RCIS.2017.7956519

[32] Ming Fan, Le Yu, Sen Chen, Hao Zhou, Xiapu Luo, Shuyue Li, Yang Liu, Jun Liu, and Ting Liu. 2020. An Empirical Evaluation of GDPR Compliance Violations in Android mHealth Apps. In *Proceedings of ISSRE 2020*. IEEE, 253–264. doi:10.1109/ISSRE5003.2020.00032

[33] Robert Feldt, Simon M. Poulding, David Clark, and Shin Yoo. 2016. Test Set Diameter: Quantifying the Diversity of Sets of Test Cases. In *2016 IEEE International Conference on Software Testing, Verification and Validation, ICST 2016*. IEEE Computer Society, 223–233. doi:10.1109/ICST.2016.33

[34] Nick Feng, Lina Marsso, Sinem Getir Yaman, Isobel Standen, Yesugen Baatartogtokh, Reem Ayad, Victória Oldemburgo de Mello, Beverley A. Townsend, Hanne Bartels, Ana Cavalcanti, Radu Calinescu, and Marsha Chechik. 2024. Normative Requirements Operationalization with Large Language Models. In *32nd IEEE International Requirements Engineering Conference, RE 2024*. IEEE, 129–141. doi:10.1109/RE59067.2024.00022

[35] Kasra Ferdowsi, Ruanqianqian (Lisa) Huang, Michael B. James, Nadia Polikarpova, and Sorin Lerner. 2024. Validating AI-Generated Code with Live Programming. In *Conference on Human Factors in Computing Systems, CHI 2024*. ACM, 143:1–143:8. doi:10.1145/3613904.3642495

[36] Sepideh Ghanavati, Daniel Amyot, and André Rifaut. 2014. Legal goal-oriented requirement language (legal GRL) for modeling regulations. In *Proceedings of MiSE 2014*. ACM, 1–6. doi:10.1145/2593770.2593780

[37] Michaela Greiler, Arie van Deursen, and Andy Zaidman. 2012. Measuring Test Case Similarity to Support Test Suite Understanding. In *Objects, Models, Components, Patterns - 50th International Conference, TOOLS 2012 (LNCS, Vol. 7304)*. Springer, 91–107. doi:10.1007/978-3-642-30561-0_8

[38] Khalid El Haji, Carolin E. Brandt, and Andy Zaidman. 2024. Using GitHub Copilot for Test Generation in Python: An Empirical Study. In *5th ACM/IEEE*

International Conference on Automation of Software Test (AST 2024). ACM, 45–55. doi:10.1145/3644032.3644443

[39] Majid Hatamian, Samuel Wairimu, Nurul Momen, and Lothar Fritsch. 2021. A privacy and security analysis of early-deployed COVID-19 contact tracing Android apps. *Empir. Softw. Eng.* 26, 3 (2021), 36. doi:10.1007/S10664-020-09934-4

[40] Silvia Ingolfo, Alberto Siena, John Mylopoulos, Angelo Susi, and Anna Perini. 2013. Arguing regulatory compliance of software requirements. *Data Knowl. Eng.* 87 (2013), 279–296. doi:10.1016/J.DATAK.2012.12.004

[41] Evelyn Kempe and Aaron Massey. 2021. Perspectives on Regulatory Compliance in Software Engineering. In *29th IEEE International Requirements Engineering Conference, RE 2021*. IEEE, 46–57. doi:10.1109/RE51729.2021.00012

[42] Kevin Kolyakov, Lina Marsso, Nick Feng, Junwei Quan, and Marsha Chechik. 2025. LEGOS-SLEEC: Tool for Formalizing and Analyzing Normative Requirements. In *47th IEEE/ACM International Conference on Software Engineering, ICSE 2025 - Companion*. IEEE, 33–36. doi:10.1109/ICSE-COMPANION66252.2025.00018

[43] Oleksandr Kosenkov, Parisa Elahidoost, Tony Gorschek, Jannik Fischbach, Daniel Mendez, Michael Unterkalmsteiner, Davide Fucci, and Rahul Mohanani. 2025. Systematic mapping study on requirements engineering for regulatory compliance of software systems. *Inf. Softw. Technol.* 178 (2025), 107622. doi:10.1016/J.INFSOF.2024.107622

[44] Kangcheng Luo, Quzhe Huang, Cong Jiang, and Yansong Feng. 2025. Automating Legal Interpretation with LLMs: Retrieval, Generation, and Evaluation. In *63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025*. Association for Computational Linguistics, 4015–4047. doi:10.18653/V1/2025.ACL-LONG.204

[45] Lukas Madl, Soner Bargu, and Mert Cuhadaroglu. 2025. Bridging Ethics and Regulation: How VBE Facilitates Compliance with the EU AI Act in High-Risk and General Purpose AI. In *Digital Humanism - First Interdisciplinary Science and Research Conference, DIGHUM 2025 (LNCS, Vol. 16319)*. Springer, 203–218. doi:10.1007/978-3-032-11108-1_15

[46] Colleen McClain, Michelle Faverio, Monica Anderson, and Eugenie Park. 2023. How Americans view data privacy. *Pew Research Center* (2023).

[47] Mashal Afzal Memon, Gian Luca Scoccia, Marco Autili, and Paola Inverardi. 2024. An Architecture for Ethics-Based Negotiation in the Decision-Making of Intelligent Autonomous Systems. In *21st IEEE International Conference on Software Architecture, ICSA 2024 - Companion, Hyderabad, India, June 4-8, 2024*. IEEE, 60–64. doi:10.1109/ICSA-C63560.2024.00016

[48] Breno Miranda, Emilio Cruciani, Roberto Verdecchia, and Antonia Bertolino. 2018. FAST approaches to scalable similarity-based test case prioritization. In *40th International Conference on Software Engineering, ICSE 2018*. ACM, 222–232. doi:10.1145/3180155.3180210

[49] Mohammed Mubarkoot, Jörn Altmann, Morteza Rasti Barzoki, Bernhard Egger, and Hyejin Lee. 2023. Software Compliance Requirements, Factors, and Policies: A Systematic Literature Review. *Comput. Secur.* 124 (2023), 102985. doi:10.1016/J.COSE.2022.102985

[50] Keqiang Pan, Xin Shen, and Shuwen Li. 2026. A Utility-Aware Probabilistic Digital Twin Framework With CC-MPC for Hard-Compliance Real-Time Decision Support in Continuous-Flow Environmental Treatment. *IEEE Access* 14, 71678–71689. doi:10.1109/ACCESS.2026.3691971

[51] Harshvardhan J. Pandit, Declan O'Sullivan, and Dave Lewis. 2019. Test-Driven Approach Towards GDPR Compliance. In *Semantic Systems. The Power of AI and Knowledge Graphs - 15th International Conference, SEMANTiCS 2019 (LNCS, Vol. 11702)*. Springer, 19–33. doi:10.1007/978-3-030-33220-4_2

[52] Alireza Parvizimosaed, Sepehr Sharifi, Daniel Amyot, Luigi Logrippo, Marco Roveri, Aidin Rasti, Ali Roudak, and John Mylopoulos. 2022. Specification and analysis of legal contracts with Symboleo. *Softw. Syst. Model.* 21, 6 (2022), 2395–2427. doi:10.1007/S10270-022-01053-6

[53] Krishna Ronanki, Christian Berger, and Jennifer Horkoff. 2023. Investigating ChatGPT's Potential to Assist in Requirements Elicitation Processes. In *49th Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2023*. IEEE, 354–361. doi:10.1109/SEAA60479.2023.00061

[54] Teresa Scantamburlo, Paolo Falcarin, Alberto Veneri, Alessandro Fabris, Chiara Gallese, Valentina Billa, Francesca Rotolo, and Federico Marcuzzi. 2024. Software Systems Compliance with the AI Act: Lessons Learned from an International Challenge. In *2nd International Workshop on Responsible AI Engineering, RAIE 2024*. ACM, 44–51. doi:10.1145/3643691.3648589

[55] Martina Siclari, Salomé Lannier, Olivier Voordeckers, Stanislaw Tosza, Sallam Abualhaija, Marcello Ceci, Nicolas Sannier, and Domenico Bianculli. 2025. Beyond silos: Bridging the gap between law and software engineering-Challenges, successes, and lesson drawing. *Internet Policy Review* 14, 4 (2025). doi:10.14763/2025.4.2042

[56] Anmol Singhal and Travis D. Breaux. 2025. Legal Requirements Translation from Law. In *33rd IEEE International Requirements Engineering Conference, RE 2025*. IEEE, 205–217. doi:10.1109/RE63999.2025.00028

[57] Amin Sleimi, Marcello Ceci, Mehrdad Sabetzadeh, Lionel C. Briand, and John Dann. 2020. Automated Recommendation of Templates for Legal Requirements. In *28th IEEE International Requirements Engineering Conference, RE 2020*. IEEE, 158–168. doi:10.1109/RE48521.2020.00027

[58] Amin Sleimi, Nicolas Sannier, Mehrdad Sabetzadeh, Lionel C. Briand, Marcello Ceci, and John Dann. 2021. An automated framework for the extraction of semantic legal metadata from legal texts. *Empir. Softw. Eng.* 26, 3 (2021), 43. doi:10.1007/S10664-020-09933-5

[59] Francesco Sovrano, Michaël Lognoul, and Alberto Bacchelli. 2024. An Empirical Study on Compliance with Ranking Transparency in the Software Documentation of EU Online Platforms. In *46th International Conference on Software Engineering: Software Engineering in Society, ICSE-SEIS2024*. ACM, 46–56. doi:10.1145/3639475.3640112

[60] Andreas Steffens, Horst Lichter, and Marco Moscher. 2018. Towards Data-driven Continuous Compliance Testing. In *Combined Proceedings of the Workshops of the German Software Engineering Conference 2018 (SE 2018) (CEUR Workshop Proceedings, Vol. 2066)*. CEUR-WS.org, 78–84. https://ceur-ws.org/Vol-2066/cse2018paper02.pdf

[61] Zeya Tan and Wei Song. 2023. PTPDroid: Detecting Violated User Privacy Disclosures to Third-Parties of Android Apps. In *45th IEEE/ACM International Conference on Software Engineering, ICSE 2023*. IEEE, 473–485. doi:10.1109/ICSE48619.2023.00050

[62] Alexander Tarlinder. 2016. *Developer Testing: Building Quality Into Software*. Addison-Wesley Professional.

[63] The European Parliament and the Council of the European Union. 2016. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). https://eur-lex.europa.eu/eli/reg/2016/679/oj

[64] The European Parliament and the Council of the European Union. 2024. AI Act–Regulation (EU) 2024/1689 of the European Parliament and of the Council of 13 June 2024 Laying Down Harmonised Rules on Artificial Intelligence and Amending Regulations (EC) No 300/2008, (EU) No 167/2013, (EU) No 168/2013, (EU) 2018/858, (EU) 2018/1139 and (EU) 2019/2144 and Directives 2014/90/EU, (EU) 2016/797 and (EU) 2020/1828 (Artificial Intelligence Act).

[65] Jake Tom, Eduard Sing, and Raimundas Matulevicius. 2018. Conceptual Representation of the GDPR: Model and Application Directions. In *Proceedings of BIR (Lecture Notes in Business Information Processing, Vol. 330)*. Springer, 18–28. doi:10.1007/978-3-319-99951-7_2

[66] Damiano Torre, Mauricio Alférez, Ghanem Soltana, Mehrdad Sabetzadeh, and Lionel C. Briand. 2021. Modeling data protection and privacy: application and experience with GDPR. *Softw. Syst. Model.* 20, 6 (2021), 2071–2087. doi:10.1007/S10270-021-00935-5

[67] Muhammad Usman, Michael Felderer, Michael Unterkalmsteiner, Eriks Klotins, Daniel Méndez, and Emil Alégroth. 2020. Compliance Requirements in Large-Scale Software Development: An Industrial Case Study. In *21st International Conference on Product-Focused Software Process Improvement (LNCS, Vol. 12562)*. Springer, 385–401. doi:10.1007/978-3-030-64148-1_24

[68] Matthias Wagner, Marie-Therese Wittmann, and Markus Borg. 2025. A Critical Look at the EU AI Act's Requirements and Affected Systems - Who Must Explain What?. In *33rd IEEE International Requirements Engineering Conference, RE 2025*. IEEE, 494–503. doi:10.1109/REW66121.2025.00074

[69] Zhiyi Xue, Xiaohong Chen, and Min Zhang. 2026. Explicating Tacit Regulatory Knowledge from LLMs to Auto-Formalize Requirements for Compliance Test Case Generation. *CoRR* abs/2601.09762 (2026). arXiv:2601.09762 doi:10.48550/ARXIV.2601.09762

[70] Rahulkrishna Yandrapally, Saurabh Sinha, Rachel Tzoref-Brill, and Ali Mesbah. 2023. Carving UI Tests to Generate API Tests and API Specification. In *45th IEEE/ACM International Conference on Software Engineering, ICSE 2023*. IEEE, 1971–1982. doi:10.1109/ICSE48619.2023.00167

[71] Zhenzhen Yang, Rubing Huang, Chenhui Cui, Nan Niu, and Dave Towey. 2025. Requirements-Based Test Generation: A Comprehensive Survey. *CoRR* abs/2505.02015 (2025). arXiv:2505.02015 doi:10.48550/ARXIV.2505.02015

[72] Nicola Zeni, Nadzeya Kiyavitskaya, Luisa Mich, James R. Cordy, and John Mylopoulos. 2015. GaiusT: supporting the extraction of rights and obligations for regulatory compliance. *Requir. Eng.* 20, 1 (2015), 1–22. doi:10.1007/S00766-013-0181-8

[73] Kaifa Zhao, Xian Zhan, Le Yu, Shiyao Zhou, Hao Zhou, Xiapu Luo, Haoyu Wang, and Yepang Liu. 2023. Demystifying Privacy Policy of Third-Party Libraries in Mobile Apps. In *45th IEEE/ACM International Conference on Software Engineering, ICSE 2023*. IEEE, 1583–1595. doi:10.1109/ICSE48619.2023.00137