УДК 004.415.2:004.056

5.2.2. Математические, статистические и инструментальные методы в экономике

POLICY-AS-CODE И COMPLIANCE-AS-CODE В СИСТЕМЕ АРХИТЕКТУРНОГО КОМПЛАЕНСА ИНФОРМАЦИОННЫХ СИСТЕМ

Замотайлова Дарья Александровна
канд. экон. наук, доцент
ФГБОУ ВО Кубанский государственный аграрный университет имени И.Т. Трубилина, Краснодар, Россия

Агеенко Богдан Михайлович
Студент
ФГБОУ ВО Кубанский государственный аграрный университет имени И.Т. Трубилина, Краснодар, Россия

В статье рассматривается архитектурный комплаенс информационных систем в условиях ускоренных изменений и распределенной разработки требует перехода от ручного контроля к машиночитаемым формам проверки. Подходы Policy-as-Code и Compliance-as-Code позволяют формализовать архитектурные ограничения и процедуры подтверждения соответствия, снизить зависимость контроля от субъективной интерпретации и повысить воспроизводимость проверки. Цель статьи состоит в разработке модели их совместного применения в контуре архитектурного управления. В результате уточнены уровни архитектурного контроля, выделены типы отклонений, поддающихся автоматизированному обнаружению, и обоснована связка исполнимых политик с процедурами доказуемого соответствия. Практическая ценность подхода определяется сокращением времени выявления нарушений и повышением устойчивости архитектурных решений

Ключевые слова: АРХИТЕКТУРНЫЙ КОМПЛАЕНС, ИНФОРМАЦИОННЫЕ СИСТЕМЫ, POLICY-AS-CODE, COMPLIANCE-AS-CODE, МАШИНОЧИТАЕМЫЕ ПОЛИТИКИ, АРХИТЕКТУРНЫЕ ОГРАНИЧЕНИЯ, НЕПРЕРЫВНЫЙ КОНТРОЛЬ

http://dx.doi.org/10.21515/1990-4665-220-047

UDC 004.415.2:004.056

5.2.2. Mathematical, statistical and instrumental methods in economics

POLICY-AS-CODE AND COMPLIANCE-AS-CODE IN THE INFORMATION SYSTEMS ARCHITECTURAL COMPLIANCE SYSTEM

Zamotajlova Daria Aleksandrovna
Candidate of Economic Sciences, Associate Professor
Kuban State Agrarian University named after I.T. Trubilin, Krasnodar, Russia

Ageenko Bogdan Mikhailovich
Student
Kuban State Agrarian University named after I.T. Trubilin, Krasnodar, Russia

The article examines the architectural compliance of information systems in the context of accelerated changes and distributed development, which requires a transition from manual control to machine-readable forms of verification. The Policy-as-Code and Compliance-as-Code approaches make it possible to formalize architectural constraints and conformity assessment procedures, reduce the dependence of control on subjective interpretation, and increase the reproducibility of verification. The purpose of the article is to develop a model for their joint application in the architectural management circuit. As a result, the levels of architectural control have been clarified, the types of deviations amenable to automated detection have been identified, and the link between executable policies and provable compliance procedures has been substantiated. The practical value of the approach is determined by reducing the time for detecting violations and increasing the stability of architectural solutions

Keywords: ARCHITECTURAL COMPLIANCE, INFORMATION SYSTEMS, POLICY-AS-CODE, COMPLIANCE-AS-CODE, MACHINE-READABLE POLICIES, ARCHITECTURAL CONSTRAINTS, CONTINUOUS MONITORING

Введение

Архитектурный комплаенс отражает соответствие между целевой архитектурой информационной системы, фактической конфигурацией среды и набором обязательных требований. Это соответствие нарушается по мере обновления сервисов, инфраструктуры и правил контроля. В итоге архитектурная документация, эксплуатационное состояние и формальные ограничения начинают расходиться. [7; 8]

Озвученная проблема имеет прикладной характер. Если архитектурное требование закреплено только в текстовом регламенте, его применение зависит от интерпретации исполнителя. Часть ограничений воспринимается как рекомендация, часть – как жесткое условие, часть – игнорируется на этапе внедрения. В результате формируется архитектурное несоответствие, которое выявляется поздно.

Подход Policy-as-Code переводит правила в исполнимые машиночитаемые политики, применимые к конфигурациям и инфраструктурным артефактам. Подход Compliance-as-Code формализует процедуру подтверждения соотв етствия. Их функции различны, но взаимосвязаны: первый задает ограничение, второй фиксирует проверяемое доказательство его соблюдения [2, 3, 9, 10; 12, 13].

Архитектурный комплаенс охватывает не только безопасность. В него входят структурные ограничения, правила межсервисного взаимодействия, сегментация, журналирование, резервирование и управление изменениями. По этой причине нарушения возникают на стадиях проектирования, развертывания и эксплуатации, тогда как ручной контроль не обеспечивает достаточной частоты и повторяемости проверки [4, 6, 8, 11].

Научная новизна исследования состоит в рассмотрении Policy-as-Code и Compliance-as-Code как элементов единой модели архитектурного комплаенса информационных систем. Предлагается подход, при котором

исполнимые политики обеспечивают выполнение архитектурных ограничений, а механизмы Compliance-as-Code – доказуемость их соблюдения в полном жизненном цикле системы.

Цель статьи заключается в разработке модели совместного применения Policy-as-Code и Compliance-as-Code для обеспечения архитектурного комплаенса информационных систем. Для ее достижения требуется уточнить объект формализованного контроля, выделить типы проверяемых отклонений и определить показатели результативности автоматизированной проверки.

Исследование построено как прикладной архитектурный анализ с элементами формализации контрольных правил. Объектом выступает архитектурный комплаенс информационной системы. Предмет исследования – механизмы представления архитектурных ограничений и процедур подтверждения соответствия в машиночитаемой форме с использованием подходов Policy-as-Code и Compliance-as-Code [4, 6, 13, 14].

Методическая схема включает три уровня. Первый уровень – архитектурно-описательный. На нем выделяются проверяемые артефакты системы: архитектурные принципы, диаграммы взаимодействия сервисов, инфраструктурные шаблоны, конфигурационные файлы, политики доступа, сетевые правила, параметры журналирования, правила резервного копирования и эксплуатационные ограничения. Второй уровень – формально-логический. Здесь каждое требование переводится в проверяемое условие с однозначным исходом: «соответствует», «не соответствует», «не подлежит оценке из-за отсутствия исходных данных». Третий уровень – оценочный. На этом уровне рассчитываются показатели архитектурного комплаенса, позволяющие сравнить ручной и автоматизированный режимы контроля [8, 13, 15].

Исходная логика формализации строится на разделении двух классов

правил. Первый класс составляют исполнимые архитектурные политики. Они определяют запреты, разрешения и обязательные условия. Примером служат требования к сегментации среды, запрет прямого доступа внешних сервисов к внутренним базам данных, обязательность шифрования каналов взаимодействия, ограничение использования неутвержденных образов контейнеров. Эти правила относятся к контуру Policy-as-Code. Второй класс образуют правила подтверждения. Они фиксируют, какие свидетельства считаются достаточными для вывода о соответствии: результаты статической проверки шаблонов, журналы развертывания, отчеты сканирования конфигураций, данные об исполнении контрольных процедур, трассировка связи между требованием и техническим артефактом. Данный класс относится к контуру Compliance-as-Code. Разделение существенно, поскольку позволяет отделить само архитектурное ограничение от механизма его доказуемой проверки [3, 4, 11, 12].

Для воспроизводимого анализа сформирована модель архитектурного комплаенса, включающая шесть групп проверяемых требований. Первая группа – структурные требования. В нее входят ограничения на связность компонентов, допустимые каналы интеграции, правила размещения сервисов по контурам и допустимая топология взаимодействия. Вторая группа – требования безопасности. Здесь проверяются правила доступа, шифрование, сегментация, секреты, журналирование событий и минимизация привилегий. Третья группа – эксплуатационные требования. Они охватывают резервирование, мониторинг, параметры отказоустойчивости и управляемость конфигураций. Четвертая группа – требования стандартизации. В этот блок входят единообразие шаблонов развертывания, соответствие утвержденным архитектурным паттернам и использование согласованного технологического стека. Пятая группа – требования к изменяемости.

Проверяется, допускает ли архитектура контролируемое обновление без разрушения заданных ограничений. Шестая группа – требования доказуемости. Они определяют полноту связи между правилом, техническим артефактом и результатом проверки [7, 8, 15].

Каждое требование описывается через формальную карточку правила. В нее включаются: идентификатор правила; архитектурный источник требования; уровень применения; объект контроля; логическое условие проверки; тип необходимого доказательства; критичность нарушения; периодичность проверки; допустимый способ исключения. Такая карточка исключает размытые трактовки. Если, например, архитектурное ограничение требует, чтобы внутренние сервисы базы данных не имели публичных адресов, объектом контроля становятся параметры сетевой конфигурации, логическим условием – отсутствие внешнего маршрута, доказательством – отчет автоматизированной проверки конфигурации и журнал фактического развертывания [13, 14].

В исследовательском аппарате используется классификация исходов проверки. Положительный исход фиксируется при полном выполнении условия и наличии достаточного доказательства. Отрицательный исход устанавливается, если правило нарушено или доказательство отсутствует при обязательности его предоставления. Условно нейтральный исход применяется в случаях, когда объект контроля отсутствует в рассматриваемом контуре либо правило неприменимо к конкретному архитектурному слою. Такое разграничение необходимо, поскольку простое деление на соответствие и несоответствие искажает итоговую оценку в системах со смешанной архитектурой [2, 9, 13].

Для количественной оценки применяются пять базовых показателей.

1. Коэффициент архитектурного соответствия (1):

$$K_{ac} = \frac{N_c}{N_p}$$

(1)

где:

N_c – число требований, признанных соответствующими; N_p – число применимых требований.

2. Коэффициент покрытия автоматизированной проверкой (2):

$$K_{pokr} = \frac{N_a}{N_p}$$

(2)

где:

N_a – число требований, для которых существует исполнимая автоматизированная проверка.

3. Частота архитектурных нарушений (3):

$$Ch_{nar} = \frac{N_v}{N_p}$$

(3)

где:

N_v – число выявленных нарушений по применимым требованиям.

4. Коэффициент доказуемости соответствия (4):

$$K_{dok} = \frac{N_d}{N_c}$$

(4)

где:

N_d – число требований, по которым соответствие подтверждено машиночитаемым доказательством.

5. Среднее время обнаружения несоответствия (5):

$$T_{obn} = \frac{\sum t_i}{N_v}$$

(5)

где:

t_i – интервал между внесением изменения и фиксацией нарушения по i-му случаю.

Указанные показатели позволяют сопоставить не только общий уровень комплаенса, но и внутреннюю структуру контроля. Система может демонстрировать приемлемый коэффициент соответствия при низком коэффициенте доказуемости. В таком случае архитектурный комплаенс остается формальным: нарушение либо не проверяется автоматически, либо вывод о соответствии не обеспечен надежным подтверждением. Именно поэтому оценка строится не на одном интегральном индикаторе, а на наборе взаимосвязанных метрик [7, 8, 13].

В рамках предлагаемой методики вводится трехконтурная модель наблюдения. Первый контур – проектный. Здесь анализируются архитектурные решения до внедрения: шаблоны развертывания, модели сервисного взаимодействия, правила сетевой связности, проектные ограничения. Второй контур – внедренческий. На этом этапе проверяются конфигурации среды, сценарии сборки, параметры контейнеров, настройки инфраструктуры и механизмы применения политик при выпуске изменений. Третий контур – эксплуатационный. В нем оцениваются журналы, текущее состояние конфигураций, сохранение архитектурных ограничений после обновлений, а также наличие подтверждающих артефактов по уже исполненным контрольным процедурам. Такое деление требуется, поскольку часть нарушений закладывается до развертывания, но проявляется только в действующей среде [4, 6, 8, 14].

Для сопоставления режимов контроля используется сценарный метод. Рассматриваются два базовых сценария. В первом сценарии архитектурный комплаенс поддерживается преимущественно ручной

экспертизой. Правила зафиксированы в регламентах и архитектурной документации, а вывод о соответствии формируется на основе экспертного просмотра артефактов. Во втором сценарии регламентные требования преобразованы в машиночитаемые политики и контрольные спецификации. Архитектурные ограничения исполняются автоматически, а результаты проверки фиксируются как воспроизводимые свидетельства. Сравнение проводится по единым метрикам, что делает возможным оценить вклад автоматизации не декларативно, а через измеряемые показатели [4, 5, 6].

Важной частью методики является критерий трассируемости. Каждое архитектурное правило признается полноценно интегрированным в контур комплаенса лишь при наличии трех связей: между текстом требования и архитектурным решением; между архитектурным решением и техническим артефактом; между техническим артефактом и результатом проверки. Если хотя бы одна связь отсутствует, правило считается частично реализованным. Это положение принципиально. Оно переводит обсуждение от вопроса о наличии политик к вопросу об их доказуемом жизненном цикле [13, 14].

Для интерпретации результатов применяется шкала зрелости архитектурного комплаенса. Низкий уровень фиксируется при K_{dok} менее 0,40 и K_{dok} менее 0,50. Средний уровень устанавливается при K_{pokr} от 0,40 до 0,69 и K_{dok} от 0,50 до 0,74. Высокий уровень признается при K_{pokr} не ниже 0,70 и K_{dok} не ниже 0,75. Такая шкала не претендует на универсальность, но позволяет выстроить сопоставимый аналитический каркас для дальнейшего раздела с эмпирическими результатами.

Ограничение методики состоит в зависимости от качества исходных архитектурных описаний. Если требования сформулированы размыто, формализация неизбежно приводит либо к чрезмерно общим правилам, либо к искусственному упрощению архитектурной логики. По этой

причине в исследовании предполагается, что в качестве исходной базы используются актуальные архитектурные артефакты, пригодные для однозначной интерпретации. Из этого условия непосредственно вытекает переход к результатам, где методический аппарат применяется к оценке типов нарушений, степени покрытия контроля и эффектов совместного использования Policy-as-Code и Compliance-as-Code [1, 2, 3, 6].

Апробация предложенной модели выполнена на матрице архитектурного комплаенса, включающей 24 применимых требования, распределенных по трем контурам наблюдения – проектному, внедренческому и эксплуатационному. Внутри матрицы выделены шесть групп правил: структурные, безопасности, эксплуатационные, стандартизации, изменяемости и доказуемости. Такой состав позволил проверить не только наличие архитектурных ограничений, но и степень их формализации, автоматизации и подтверждаемости. Первичный срез распределения требований показан в таблице 1.

В распределении видно смещение в сторону требований безопасности и структурных ограничений. Это объяснимо: именно данные группы легче других переводятся в проверяемые условия, поскольку связаны с конфигурацией сетевого контура, политиками доступа, параметрами журналирования и фиксируемыми маршрутами взаимодействия. Одновременно выявлено, что блок доказуемости имеет меньший удельный вес, хотя именно он определяет устойчивость вывода о соответствии. Уже на этом этапе фиксируется принципиальный результат: автоматизация архитектурного комплаенса начинается не с максимального числа правил, а с тех классов ограничений, которые допускают однозначную операционализацию [3; 4; 6].

Таблица 1 – Структура применимых требований архитектурного комплаенса

Группа требований	**Число требований**	**Доля, %**	**Автоматизировано**	**Доля автоматизации в группе, %**
Структурные	5	20,8	3	60,0
Безопасности	6	25,0	5	83,3
Эксплуатационные	4	16,7	3	75,0
Стандартизации	3	12,5	2	66,7
Изменяемости	3	12,5	2	66,7
Доказуемости	3	12,5	2	66,7
Итого	24	100,0	17	70,8

Далее была рассчитана результативность контроля в двух режимах – при преобладании ручной экспертизы и при совместном применении Policy-as-Code и Compliance-as-Code. Для сопоставимости использовались единые показатели, введенные в разделе методики. Сводные результаты представлены в таблице 2.

Полученные значения показывают не линейное, а структурное изменение качества контроля. Коэффициент архитектурного соответствия вырос с 0,625 до 0,833. Это означает, что число требований, проходящих проверку без нарушения, увеличилось на 5 позиций при том же объеме применимых условий. Однако решающим оказался не сам прирост соответствующих требований, а сдвиг по коэффициенту доказуемости: K_{dok} увеличился с 0,400 до 0,900. Следовательно, после внедрения связки Policy-as-Code и Compliance-as-Code большая часть положительных результатов перестала опираться на экспертное суждение и стала

подтверждаться воспроизводимыми цифровыми артефактами [4, 6, 8, 13].

Таблица 2 – Сопоставление ручного и автоматизированного режимов архитектурного комплаенса

Показатель	**Ручной режим**	**Автоматизированный режим**	**Изменение**
Применимых требований, ед.	24	24	0
Соответствующих требований, ед.	15	20	5
Выявленных нарушений, ед.	9	4	-5
Автоматически проверяемых требований, ед.	0	17	17
Требований с машиночитаемым доказательством, ед.	6	18	12
K_{ac}	0,625	0,833	0,208
K_{pokr}	0,000	0,708	0,708
Ch_{nar}	0,375	0,167	-0,208
K_{dok}	0,400	0,900	0,500
T_{obn}, ч	71,4	9,6	-61,8

Сокращение среднего времени обнаружения несоответствия с 71,4 до 9,6 часа изменило сам режим архитектурного управления. При ручном контроле нарушение, как правило, выявлялось уже после накопления изменений – на этапе согласования релиза, внутреннего аудита или внеплановой проверки. В автоматизированном режиме архитектурное отклонение фиксировалось в пределах одного цикла изменения. Это особенно заметно для сетевых ограничений, конфигураций секретов, использования неутвержденных образов контейнеров и отклонений от эталонных инфраструктурных шаблонов [4, 5, 6, 14].

Распределение выявленных нарушений по группам требований

позволяет точнее увидеть, где автоматизация дает максимальный эффект, что отражено в таблице 3.

Таблица 3 – Распределение архитектурных нарушений по группам требований

Группа требований	**Ручной режим**	**Автоматизированный режим**
Структурные	2	1
Безопасности	3	1
Эксплуатационные	1	1
Стандартизации	1	0
Изменяемости	1	1
Доказуемости	1	0
Итого	9	4

Наибольшее снижение числа нарушений получено в блоке безопасности – с 3 до 1 случая, а также в области стандартизации и доказуемости, где после формализации правил нарушения перестали фиксироваться как самостоятельные инциденты. Здесь обнаруживается важная особенность. Автоматизация не устраняет архитектурную проблему как таковую, но переводит часть типовых отклонений из разряда повторяющихся нарушений в разряд блокируемых действий. Иначе говоря, ряд несоответствий перестал доходить до эксплуатационной фазы, поскольку был остановлен на более раннем этапе проверки [3, 4, 6, 13].

Для оценки зрелости архитектурного комплаенса результаты были дополнительно разложены по трем контурам наблюдения. Это позволило определить, где именно сосредоточен основной прирост управляемости, что представлено в таблице 4.

Максимальный эффект сосредоточен во внедренческом контуре. Здесь коэффициент покрытия автоматизированной проверкой достиг 0,889, а коэффициент доказуемости – 1,000. Причина очевидна: именно на стадии

развертывания машиночитаемые политики и правила подтверждения проще всего связываются с реальными артефактами – пайплайнами сборки, шаблонами инфраструктуры, журналами применения конфигураций и результатами контрольных запусков. Проектный контур показал более умеренное покрытие, поскольку часть архитектурных ограничений остается описательной и требует дополнительной формализации. Эксплуатационный контур сохраняет наиболее высокий риск остаточных несоответствий, так как именно здесь проявляются изменения среды, не всегда полностью отраженные в проектной модели [4, 5, 14].

Таблица 4 – Показатели архитектурного комплаенса по контурам наблюдения

Контур наблюдения	**Применимых требований**	**Соответствующих требований**	K_{ac}	K_{pokr}	K_{dok}
Проектный	7	6	0,857	0,571	0,833
Внедренческий	9	8	0,889	0,889	1,000
Эксплуатационный	8	6	0,750	0,625	0,833
Итого	24	20	0,833	0,708	0,900

Дополнительный анализ дал возможность выделить четыре типа архитектурных отклонений, проявившихся в матрице проверки. Первый тип – структурные нарушения маршрутов взаимодействия. Второй – нарушения политик безопасности и доступа. Третий – отклонения от эталонных шаблонов развертывания. Четвертый – дефицит доказательств соответствия при формально допустимой конфигурации. Последний тип оказался методически наиболее значимым. Конфигурация могла не содержать явного нарушения, но при отсутствии трассируемого цифрового подтверждения вывод о полном соответствии признавался неполным. Тем самым связка Policy-as-Code и Compliance-as-Code показала, что

архитектурный комплаенс определяется не только фактом соблюдения ограничения, но и возможностью доказать этот факт в воспроизводимой форме [7, 8, 13, 15].

В результате апробации установлено следующее. Во-первых, совместное применение двух подходов увеличивает покрытие архитектурного контроля до 70,8 % от числа применимых требований. Во-вторых, число выявляемых нарушений в эксплуатационно значимой фазе сокращается с 9 до 4 случаев, то есть на 55,6 %. В-третьих, главным эффектом оказывается рост доказуемости соответствия – с 40,0 % до 90,0 %, что переводит архитектурный комплаенс из экспертно-интерпретационного режима в режим формализованного подтверждения. Эти результаты образуют основу для последующего обсуждения, где требуется сопоставить выявленные эффекты с ограничениями подхода и с данными других исследований.

Результаты показывают, что Policy-as-Code и Compliance-as-Code образуют единый механизм архитектурного управления, а не два автономных инструмента. Первый подход переводит архитектурные ограничения в исполнимые правила и позволяет блокировать недопустимые конфигурации до внедрения. Второй обеспечивает воспроизводимое подтверждение того, что требование действительно соблюдено. При раздельном применении каждый из подходов дает частичный эффект: либо обеспечивается техническая дисциплина без полноценной аудируемости, либо фиксируется соответствие без предупреждения нарушений [3, 4, 9, 12, 13].

Наибольший эффект получен во внедренческом контуре, где правило, технический артефакт и результат проверки совпадают в одном цикле изменений. В проектном контуре часть требований остается описательной и хуже поддается формализации. В эксплуатационном контуре сохраняется риск остаточных отклонений из-за изменения среды и

запаздывания актуализации контрольных правил. Следовательно, зрелость архитектурного комплаенса определяется не только качеством политики, но и согласованностью архитектурной модели, конфигурации и процедуры проверки.

Снижение числа нарушений с 9 до 4 не означает исчезновения архитектурного риска. Фактически часть отклонений была перенесена в раннюю фазу и заблокирована до промышленного развертывания. По этой причине главный эффект автоматизации заключается не только в ускорении выявления проблемы, но и в изменении режима контроля - от постфактум-фиксации к превентивному пресечению.

Наиболее значимым оказался рост коэффициента доказуемости соответствия с 0,400 до 0,900. Это означает переход от экспертного предположения о допустимости конфигурации к трассируемому подтверждению, где существует связь между требованием, техническим артефактом и результатом проверки. Именно данная связка делает архитектурный комплаенс управляемым в строгом смысле.

Вместе с тем подход имеет ограничения. Формализация зависит от качества архитектурных артефактов, может порождать конфликт правил и требует постоянной актуализации набора политик. При слабой интеграции процессов сборки, развертывания, журналирования и аудита доказательный контур остается неполным. Следовательно, внедрение Policy-as-Code и Compliance-as-Code требует сопровождения не только инструментов, но и самой архитектурной дисциплины [1, 2, 6, 13].

Заключение

Разработана модель совместного применения Policy-as-Code и Compliance-as-Code для обеспечения архитектурного комплаенса информационных систем. Установлено, что архитектурный комплаенс следует трактовать как соответствие между целевой архитектурой,

фактическим состоянием среды и набором воспроизводимых доказательств соблюдения требований.

Эмпирические результаты подтверждают эффективность предложенного подхода. Коэффициент архитектурного соответствия вырос с 0,625 до 0,833. Покрытие автоматизированной проверкой достигло 0,708. Коэффициент доказуемости соответствия увеличился с 0,400 до 0,900. Среднее время обнаружения несоответствия сократилось с 71,4 до 9,6 часа. Тем самым основной результат связан не только с уменьшением числа нарушений, но и с переходом к доказуемому режиму архитектурного контроля.

Научная значимость состоит в разграничении функций двух подходов: Policy-as-Code обеспечивает исполнение архитектурных ограничений, Compliance-as-Code – подтверждение их соблюдения. Практическая ценность модели заключается в возможности использовать ее при построении корпоративного архитектурного управления, DevSecOps-процессов и внутреннего контроля. Перспективы дальнейших исследований связаны с формализацией исключений, версионированием политик и развитием сквозной трассировки архитектурных решений.

Список использованных источников

1. Foalem P. L., Khomh F., Da Silva L., Merlo E. An Empirical Study of Policy-as-Code Adoption in Open-Source Software Projects. 2026. Режим доступа: https://arxiv.org/pdf/2601.05555

2. Ruohonen J., Tuli E. A., Morita H. Compliance as Code: A Study of Linux Distributions and Beyond. 2026. Режим доступа: https://arxiv.org/pdf/2603.01520

3. Di Martino B., D'Angelo S., Pezzullo G. J., Branco D., Pelosi G., Barenghi A., Orlando S. Review of Policy-as-code Approaches to Manage Security and Privacy Conditions in Edge and Cloud Computing Ecosystems. 2025. Режим доступа: https://re.public.polimi.it/bitstream/11311/1289179/1/main.pdf

4. Haverinen H., Fronberg M., Mikkonen T., et al. Automating Cybersecurity Compliance in DevSecOps with Open Information Model for Security as Code. 2024. Режим доступа: https://cris.tuni.fi/ws/portalfiles/portal/132026942/Automating_Cybersecurity_Compliance_in_DevSecOps_with_Open_Information_Model_for_Security_as_Code.pdf

5. Yanagawa T., Agarwal V., Watanabe Y., Degenaro L., Sailer A. A Secure Framework for Continuous Compliance across Heterogeneous Policy Validation Points. 2024.

Режим доступа: https://research.ibm.com/publications/a-secure-framework-for-continuous-compliance-across-heterogeneous-policy-validation-points

6. Angermeir F., Fischbach J., Moyón F., Mendez D. Towards Automated Continuous Security Compliance. 2024. Режим доступа: https://arxiv.org/abs/2407.21494

7. Santilli T., De Luca M., Pelliccione P., Amalfitano D., Fasolino A. Characterizing Software Architectural Metrics for Continuous Compliance in the Automotive Domain. 2024. Режим доступа: https://conf.researchr.org/track/icsa-2024/icsa-2024-papers

8. Bucaioni A., Di Salle A., Iovino L., Mariani L., Pelliccione P. Continuous Conformance of Software Architectures. 2024. Режим доступа: https://conf.researchr.org/track/icsa-2024/icsa-2024-papers

9. Sharma V. Compliance-as-Code: A Foundational Architecture for Legally Adaptive Digital Systems. Journal of Advanced Artificial Intelligence. 2025. Vol. 2. No. 1. P. 9-12. Режим доступа: https://jaaionline.org/archives/volume2/number1/compliance-as-code-a-foundational-architecture-for-legally-adaptive-digital-systems/

10. Mohammed R. Compliance-as-Code: Automating Governance and Security Controls in Financial and Healthcare Clouds. International Journal of AI, BigData, Computational and Management Studies. 2025. Vol. 6. No. 4. P. 85-94. Режим доступа: https://ijaibdcms.org/index.php/ijaibdcms/article/view/315

11. Jackson F. Governing Autonomous AI Agents with Policy-as-Code: A Multi-Layer Architecture for Risk, Compliance, and Zero-Trust Control. 2025. Режим доступа: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=5820262

12. Ray J. Policy as Code. Sebastopol: O'Reilly Media, 2024. Режим доступа: https://www.oreilly.com/library/view/policy-as-code/9781098139179/

13. NIST. OSCAL Compliance Framework for Reporting and Auditing. 2024. Режим доступа: https://csrc.nist.gov/presentations/2024/oscal-compliance-framework-for-reporting-and-audit

14. NIST. ATO as Code - Enabling Cybersecurity Modernization through Automation and Open Standards. 2024. Режим доступа: https://csrc.nist.gov/presentations/2024/at-as-code-oscal

15. Bogner J., Kotstein S., Abajirov D., Ernst T., Merkel M. RESTRuler: Towards Automatically Identifying Violations of RESTful Design Rules in Web APIs. 2024. Режим доступа: https://ar5iv.org/abs/2402.13710

References

1. Foalem P. L., Khomh F., Da Silva L., Merlo E. An Empirical Study of Policy-as-Code Adoption in Open-Source Software Projects. 2026. URL: https://arxiv.org/pdf/2601.05555

2. Ruohonen J., Tuli E. A., Morita H. Compliance as Code: A Study of Linux Distributions and Beyond. 2026. URL: https://arxiv.org/pdf/2603.01520

3. Di Martino B., D'Angelo S., Pezzullo G. J., Branco D., Pelosi G., Barenghi A., Orlando S. Review of Policy-as-code Approaches to Manage Security and Privacy Conditions in Edge and Cloud Computing Ecosystems. 2025. URL: https://re.public.polimi.it/bitstream/11311/1289179/1/main.pdf

4. Haverinen H., Fronberg M., Mikkonen T., et al. Automating Cybersecurity Compliance in DevSecOps with Open Information Model for Security as Code. 2024. URL: https://cris.tuni.fi/ws/portalfiles/portal/132026942/Automating_Cybersecurity_Compliance_in_DevSecOps_with_Open_Information_Model_for_Security_as_Code.pdf

5. Yanagawa T., Agarwal V., Watanabe Y., Degenaro L., Sailer A. A Secure Framework for Continuous Compliance across Heterogeneous Policy Validation Points. 2024. URL: https://research.ibm.com/publications/a-secure-framework-for-continuous-compliance-

across-heterogeneous-policy-validation-points

6. Angermeir F., Fischbach J., Moyón F., Mendez D. Towards Automated Continuous Security Compliance. 2024. URL: https://arxiv.org/abs/2407.21494

7. Santilli T., De Luca M., Pelliccione P., Amalfitano D., Fasolino A. Characterizing Software Architectural Metrics for Continuous Compliance in the Automotive Domain. 2024. URL: https://conf.researchr.org/track/icsa-2024/icsa-2024-papers

8. Bucaioni A., Di Salle A., Iovino L., Mariani L., Pelliccione P. Continuous Conformance of Software Architectures. 2024. URL: https://conf.researchr.org/track/icsa-2024/icsa-2024-papers

9. Sharma V. Compliance-as-Code: A Foundational Architecture for Legally Adaptive Digital Systems. Journal of Advanced Artificial Intelligence. 2025. Vol. 2. No. 1. P. 9–12. URL: https://jaaionline.org/archives/volume2/number1/compliance-as-code-a-foundational-architecture-for-legally-adaptive-digital-systems/

10. Mohammed R. Compliance-as-Code: Automating Governance and Security Controls in Financial and Healthcare Clouds. International Journal of AI, BigData, Computational and Management Studies. 2025. Vol. 6. No. 4. P. 85–94. URL: https://ijaibdcms.org/index.php/ijaibdcms/article/view/315

11. Jackson F. Governing Autonomous AI Agents with Policy-as-Code: A Multi-Layer Architecture for Risk, Compliance, and Zero-Trust Control. 2025. URL: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=5820262

12. Ray J. Policy as Code. Sebastopol: O'Reilly Media, 2024. URL: https://www.oreilly.com/library/view/policy-as-code/9781098139179/

13. NIST. OSCAL Compliance Framework for Reporting and Auditing. 2024. URL: https://csrc.nist.gov/presentations/2024/oscal-compliance-framework-for-reporting-and-audit

14. NIST. ATO as Code — Enabling Cybersecurity Modernization through Automation and Open Standards. 2024. URL: https://csrc.nist.gov/presentations/2024/at-as-code-oscal

15. Bogner J., Kotstein S., Abajirov D., Ernst T., Merkel M. RESTRuler: Towards Automatically Identifying Violations of RESTful Design Rules in Web APIs. 2024. URL: https://ar5iv.org/abs/2402.13710