

Проверяемое автономное управление: архитектура контроля решений ИИ в киберфизических и распределённых системах

Игорь Рыбаков

В статье рассматривается архитектура SEACS (Self-Evolving Autonomous Control System), предназначенная для построения проверяемого контура автономного управления в распределённых и киберфизических системах. В отличие от подходов, где успешный вызов модели или выполнение команды трактуется как завершение задачи, SEACS разделяет наблюдение, оценку состояния, выбор решения, исполнение и последующую проверку фактического результата. Рассматриваются механизмы управления риском и доверием, гистерезиса, задержки повторных воздействий, резервных сценариев и передачи решения человеку. Отдельное внимание уделено воспроизводимым испытаниям, требованиям к данным и метрикам, а также ограничениям исследовательского прототипа.

Введение

Развитие автономных систем расширяет круг решений, которые программное обеспечение принимает без непосредственного участия оператора. Это относится к распределённым вычислительным системам, промышленной автоматизации, робототехнике, энергетике, транспортным комплексам и интеллектуальной инфраструктуре.

В классической системе автоматического управления алгоритм, как правило, работает в заранее заданном пространстве состояний и управляющих воздействий. При включении моделей машинного обучения и генеративных моделей ситуация усложняется: система способна учитывать неполный контекст, формировать

ранее не предусмотренные варианты действий и обрабатывать неоднозначные входные данные.

Поэтому инженерная задача состоит не только в повышении качества самой модели, но и в создании внешнего проверяемого контура, который определяет, допустимо ли действие, на каких данных оно основано и достигнут ли ожидаемый эффект после исполнения. В SEACS решение рассматривается не как конечный результат вычисления, а как гипотеза об управляющем воздействии, которая должна пройти последовательные проверки.

Архитектура замкнутого контура

Базовый цикл SEACS состоит из пяти стадий: наблюдение, оценка, выбор

решения, исполнение и проверка. В англоязычной записи это соответствует последовательности Observe → Evaluate → Decide → Act → Verify.

На этапе наблюдения фиксируется текущее состояние объекта и среды. На этапе оценки определяется качество входных данных, уровень риска и допустимость автоматической реакции. Затем формируется кандидатное действие, которое проходит правила допуска. После исполнения система не считает задачу завершённой: она повторно наблюдает объект и сравнивает фактическое состояние с ожидаемым (рис. 1).

Принципиально важно разделять состояние, решение, действие и фактический результат. Успешный вызов API, отправка команды или завершение функции подтверждают только факт исполнения команды, но не достижение цели управления (рис. 2).

Наблюдение и оценка состояния

Качество управляющего решения зависит как от алгоритма выбора действия, так и от качества входного состояния. Слой наблюдения должен фиксировать не только измеряемые значения, но и источник данных, временную метку, доступность источника, задержку получения информации, согласованность нескольких

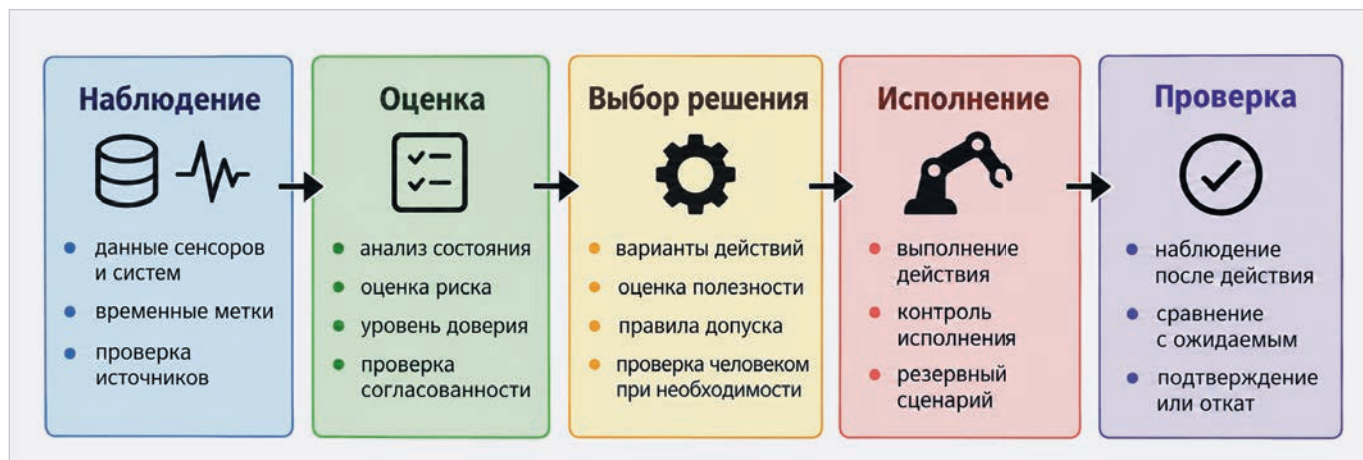


Рис. 1. Общий цикл SEACS: Наблюдение → Оценка → Выбор решения → Исполнение → Проверка

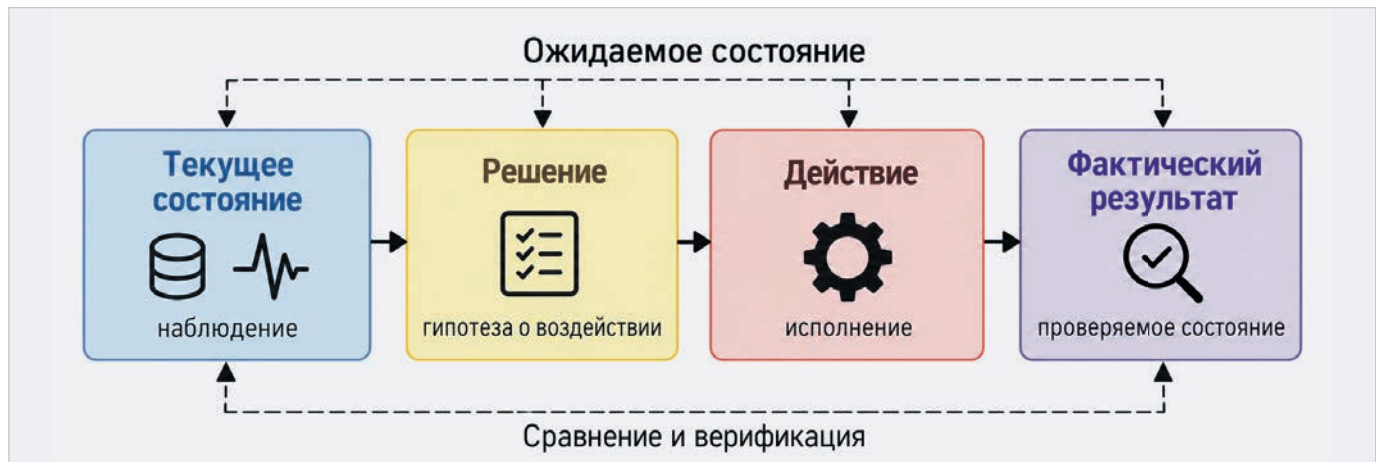


Рис. 2. Разделение состояния, решения, действия и фактического результата

источников и признаки аномального поведения.

Слой оценки преобразует поток наблюдений в формализованное состояние системы. В экспериментальной реализации SEACS используются показатели работоспособности, задержки, частоты ошибок и интегрального уровня доверия. При этом уровень доверия не трактуется как строгая статистическая вероятность истинности. Это эксплуатационная величина, определяющая, насколько безопасно действовать автоматически на основании имеющейся информации.

Такой подход позволяет разделить два вопроса: что, предположительно, происходит с системой, и насколько безопасно на основании этих данных выполнять управляющее действие. Эти ответы не всегда совпадают.

Выбор решения и управление риском

На стадии выбора решения потенциальные действия сравниваются не только по ожидаемой эффективности, но и с учётом риска. В текущем прототипе используется инженерная функция полезности вида:

полезность = ожидаемый выигрыш состояния – штраф за риск + вес доверия.

Ожидаемый выигрыш состояния отражает предполагаемое улучшение контролируемых показателей; штраф за риск учитывает возможные негативные последствия; вес доверия зависит от качества и согласованности информации. Эта формула не является универсальной математической моделью управления, а задаёт принцип ранжирования вариантов внутри

экспериментального механизма принятия решений.

Действие с максимальной полезностью не обязательно выполняется автоматически. Дополнительные правила могут запретить его выполнение или передать принятие решения человеку, если уровень доверия недостаточен, состояние системы быстро меняется, действие труднообратимо, отсутствует резервный сценарий или результат предыдущего воздействия ещё не прошёл проверку (рис. 3).

Гистерезис, задержка повторных воздействий и защита от нестабильного управления

Для автономной системы опасно не только ошибка отдельного

решения. Не менее опасно быстрое чередование формально допустимых воздействий. Если управляющий алгоритм немедленно реагирует на небольшое изменение метрики, система может перейти в режим колебаний между несколькими конфигурациями.

Для снижения этого риска в SEACS используются четыре группы механизмов:

- гистерезис – разные пороги входа в новое состояние и возврата к предыдущему;
- сглаживание – уменьшение влияния кратковременных выбросов и шумов;
- задержка повторного воздействия – временный запрет на новое действие сразу после предыдущего;

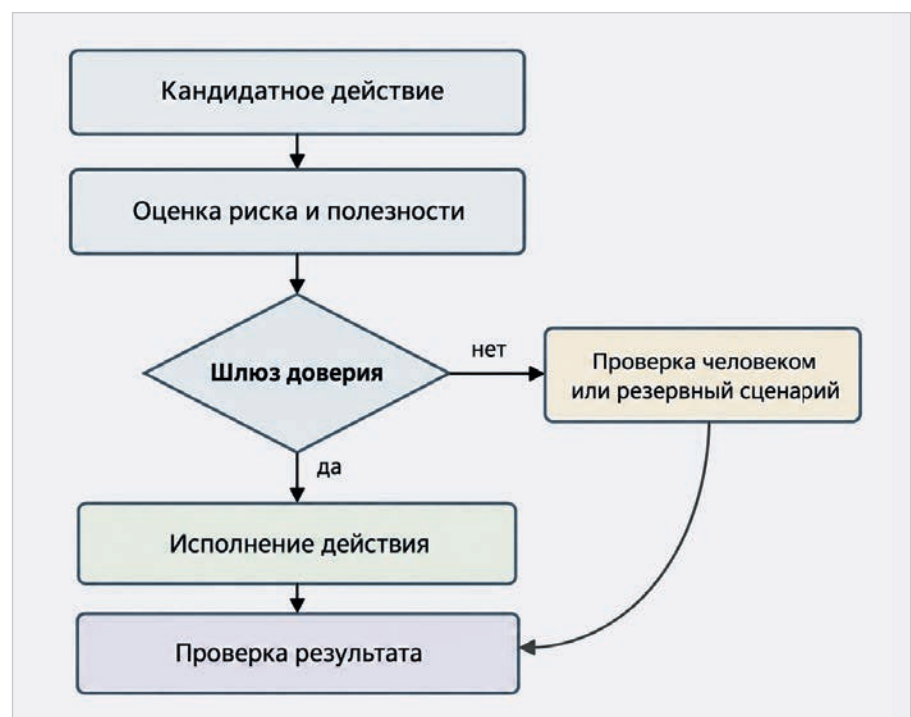


Рис. 3. Шлюз доверия: схема выбора и допуска действия



Рис. 4. Логика воспроизводимого сценария отказа и восстановления

● резервный сценарий – переход к заранее определённому безопасному варианту, если ожидаемый эффект не подтверждён.

Эти методы известны в классических системах управления и инженерии надёжности. Их роль возрастает при включении вероятностных и генеративных компонентов, поведение которых может быть менее устойчивым, чем поведение жёстко заданного автомата.

Проверка действия после исполнения

Этап проверки является одним из ключевых элементов предлагаемой архитектуры. После выполнения действия система повторно получает наблюдения и сравнивает ожидаемое состояние с фактическим.

Результат проверки может быть отнесён к одному из трёх классов: подтверждено – изменение соответ-

ствует ожидаемому эффекту; ухудшено – действие выполнено, но эффект недостаточен или возникли побочные отклонения; неуспешно – состояние ухудшилось или критические параметры вышли за допустимые границы, что требует отката или резервного сценария.

Таким образом, технически успешное исполнение команды не является доказательством успешного управления. История фактических результатов может использоваться для последующей коррекции уровня доверия к отдельным классам воздействий.

Экспериментальная проверка

Для проверки архитектуры используются воспроизводимые сценарии отказов и фиксированные тестовые прогоны. Система проверяется не только в нормальном режиме, но и при росте задержки, увеличении часто-

ты ошибок, неполных наблюдениях, конфликтующих сигналах, необходимости резервного сценария и передаче решения оператору.

Сначала задаётся логика сценария: исходное состояние, внесение отказа, изменение наблюдаемых метрик, принятие решения, выполнение действия и последующая проверка восстановления. Такой порядок позволяет отделить реакцию системы на отказ от подтверждения того, что после управляющего воздействия она действительно вернулась в допустимое состояние (рис. 4).

Для каждого прогона используется единая структура фиксации данных. В неё входят исходное состояние, внесённый отказ, полученные наблюдения, выбранное решение, условия допуска, выполненное действие, фактический результат проверки и итоговый вердикт (табл. 1).

Такая форма позволяет сравнивать разные испытания по одинаковому набору полей и не смешивать факт выполнения команды с подтверждённым результатом. В текущем

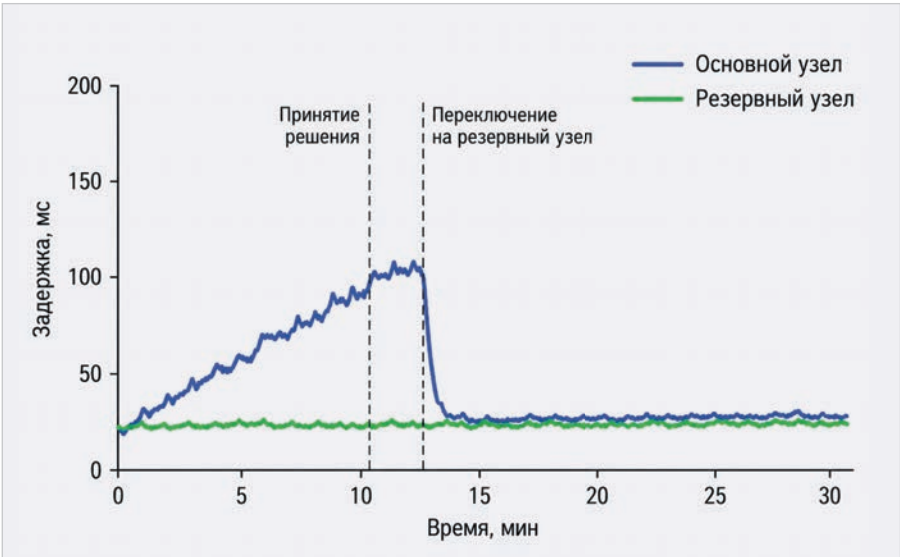


Рис. 5. Пример тестового прогона: переключение на резервный узел при росте задержки

Таблица 1. Пример структуры испытательного сценария SEACS

Поле	Содержание
Initial state	Исходное состояние
Fault	Внесённый отказ
Observation	Полученные показатели
Decision	Выбранное действие
Risk / Trust	Условия допуска
Action	Выполненное воздействие
Verification	Фактический результат
Verdict	PASS / FAIL / HUMAN REVIEW

фиксированном наборе восемь заранее определённых сценариев прошли предусмотренные проектом проверки. Этот результат следует трактовать только как прохождение собственного набора сценариев, а не как доказательство промышленной надёжности или универсальной эффективности архитектуры.

Один из демонстрационных сценариев моделирует рост задержки на основном узле распределённой системы. После достижения условий для принятия решения система переключает нагрузку на резервный узел, после чего измеряемая задержка возвращается к устойчивому уровню. Здесь проверяется не только сам факт переключения, но и достижение ожидаемого состояния после него (рис. 5).

Следующий этап экспериментальной проверки должен включать внешние сценарии, которые не использовались разработчиком при настройке логики системы.

Ограничения и инженерные компромиссы

Несмотря на потенциал подхода, внедрение SEACS в реальные системы связано с рядом ограничений. Они касаются не только зрелости программного прототипа, но и качества данных, наблюдаемости объекта, вычислительных ресурсов, безопасности и регуляторных требований.

К основным ограничениям относятся:

- качество данных – неполные, шумные или смещённые данные могут приводить к неверной оценке состояния;
- ограниченная наблюдаемость – не все внутренние состояния доступны прямому измерению, они требуют косвенной оценки;



Рис. 6. Типовая архитектура данных в SEACS

- вычислительные ресурсы – анализ сложных сценариев и применение моделей могут потребовать значительных ресурсов;
- риски безопасности – необходимо учитывать редкие, ошибочные и намеренно провоцируемые режимы;
- регуляторные и этические требования – для критических систем нужны дополнительные процедуры валидации, документирования и аудита.

На практике требуется баланс между автономностью, точностью, стоимостью вычислений, скоростью реакции и безопасностью. Для инженерных систем приоритетом должен оставаться принцип: предсказуемость и безопасность важнее максимальной производительности.

Поэтому рациональным является поэтапное внедрение: сначала

наблюдение и диагностика, затем ограниченные рекомендации, далее частичная автоматизация и только после подтверждения устойчивости – более сложные сценарии автономного поведения.

Требования к данным, метрикам и протоколам испытаний

Эффективность SEACS напрямую зависит от качества данных, корректно выбранных метрик и строгих процедур испытаний. Для инженерного применения необходимо заранее определить требования к данным, формализовать показатели качества и обеспечить воспроизводимость тестов.

Набор метрик должен охватывать как минимум надёжность, безопасность, качество управления и устойчивость. Конкретные пороговые зна-

Таблица 2. Примеры метрик и критериев приёмки для SEACS

Категория	Метрика	Описание	Типовое целевое значение
Надёжность	Вероятность отказа	Доля отказов в тестовых сценариях	$\leq 10^{-5}$ на час
	Доступность	Доля времени работоспособности	$\geq 99,9\%$
Безопасность	Частота опасных действий	Количество опасных команд на 1 час работы	$\leq 10^{-6}$
Качество управления	Среднее время реакции	Время обнаружения и реакции на аномалию	≤ 100 мс
	Отклонение от цели	Среднеквадратичное отклонение	В пределах допуска
Устойчивость	Энергетическая эффективность	Относительный расход энергии	Не хуже базового уровня
	Стабильность при возмущениях	Сохранение работоспособности при заданных отклонениях параметров	$\geq 95\%$ сценариев
	Время восстановления после сбоя	Возврат к штатному режиму	≤ 5 с

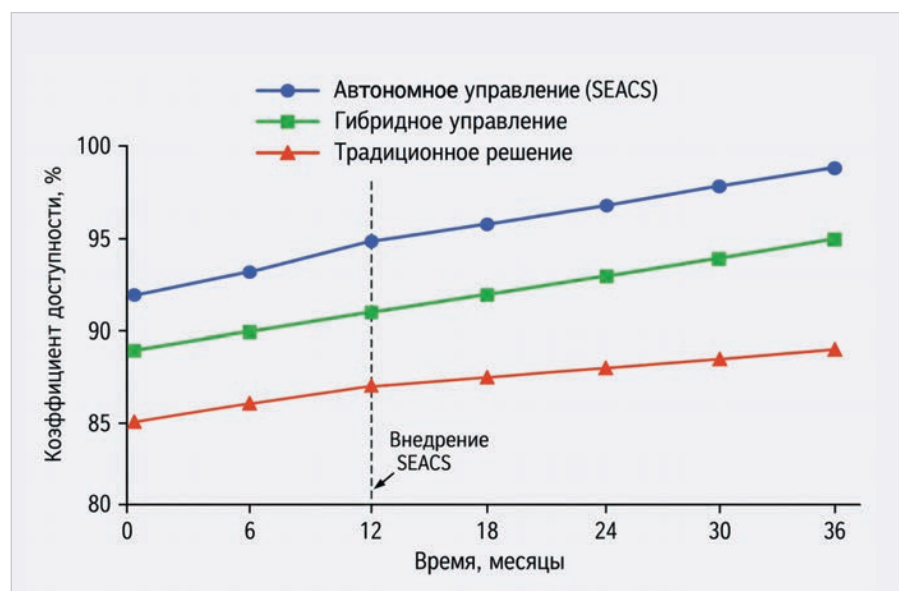


Рис. 7. Сравнение сценариев внедрения SEACS в промышленной автоматизации

чения зависят от класса системы и не должны переноситься между объектами без отдельной калибровки (табл. 2).

К данным предъявляются требования по репрезентативности режимов, происхождению и верификации источников, контролю качества, синхронизации временных рядов, обработке пропусков, хранению исторических версий и управлению разметкой.

Протокол испытаний должен включать модульные проверки отдельных функций, интеграционные испытания в контуре, сценарные тесты с отказами, валидацию на стендах и, при возможности, ограниченные испытания в реальной эксплуатации. Для критических применений дополнительно требуются процедуры независимой проверки и сертификации (рис. 6).

Практические области применения

Архитектура SEACS не привязана к одному типу оборудования. Она может быть адаптирована для разных отраслей, где требуется автономное или полуавтономное управление с контролем решений и обязательной проверкой результата.

Промышленная автоматизация:

- автономное управление производственными линиями;
- адаптация к изменению номенклатуры и условий;
- снижение простоев и энергопотребления.

Энергетические системы:

- управление распределёнными источниками энергии;

- балансировка нагрузки с учётом прогноза;
- повышение устойчивости к аварийным ситуациям.

Транспорт и робототехника:

- автономные мобильные платформы;
- координация групп роботов в динамической среде;
- безопасное взаимодействие с человеком.

Критическая инфраструктура:

- мониторинг и управление системами жизнеобеспечения;
- раннее обнаружение угроз и аномалий;
- поддержание непрерывности работы.

Аэрокосмические и оборонные применения:

- автономные аппараты и беспилотные системы;
- работа в сложных и неопределённых условиях;
- высокие требования к надёжности и безопасности.

Для иллюстрации долгосрочного эффекта можно использовать сравнительный сценарий, где автономное управление сопоставляется с гибридным и традиционным режимами. Такой график не следует трактовать как универсальный прогноз: он демонстрирует форму сравнения и место момента внедрения SEACS в жизненном цикле системы (рис. 7).

Дорожная карта развития SEACS

Развитие архитектуры предполагает поэтапное расширение функцио-

нальности, уровня автономности и степени интеграции с внешними системами. Дорожная карта должна учитывать как технические возможности, так и реальные ограничения эксплуатации.

На первом этапе приоритетом являются наблюдение, диагностика, поддержка принятия решений и интеграция с существующей инфраструктурой. На втором этапе возможна частичная автоматизация с жёсткими правилами допуска и контролем человека. На третьем этапе добавляются адаптивные и обучаемые механизмы при сохранении независимого контура проверки. Широкое применение возможно только после накопления эксплуатационных данных, стандартизации интерфейсов и подтверждения надёжности на независимых испытаниях.

Заключение

В статье рассмотрена архитектура SEACS как основа для построения проверяемого автономного управления в киберфизических и распределённых системах. Подход объединяет наблюдение, оценку состояния, выбор решения, исполнение и обязательную проверку фактического результата.

Ключевым принципом является разделение интеллектуального предложения и инженерного допуска к действию. Модель может интерпретировать события и формировать варианты реакции, но критические ограничения, правила допуска, резервные сценарии и проверка результата должны оставаться в отдельном контролируемом контуре.

Текущая реализация подтверждает техническую реализуемость подхода на фиксированном наборе воспроизводимых сценариев, однако не является доказательством промышленной готовности. Для дальнейшего развития необходимы внешние тестовые наборы, длительные эксплуатационные данные, калибровка метрик под конкретные классы объектов и независимая проверка.

Практическая ценность SEACS состоит не в замене существующей автоматической генеративной моделью, а в создании дополнительного слоя, который помогает ограничивать автономность, фиксировать причины решений и проверять, привело ли действие к ожидаемому результату.

