УДК 339.54.012

Иванов Ярослав Михайлович
Магистрант 1 курса
Университет «Уральский Государственный Университет Путей Сообщения»
Россия, г. Екатеринбург

ЭВОЛЮЦИЯ ЖИЗНЕННОГО ЦИКЛА РАЗРАБОТКИ ПО: ОТ ВОДОПАДА (WATERFALL) ЧЕРЕЗ СПИРАЛЬНУЮ МОДЕЛЬ К DEVOPS

Аннотация. В данной статье рассматривается эволюция моделей жизненного цикла разработки программного обеспечения (SDLC), отражающая стремление индустрии к повышению эффективности и качества выпускаемых продуктов. Прослеживается путь от строгой и последовательной каскадной модели (Waterfall), через итеративный подход спиральной модели с акцентом на анализ рисков, к гибким методологиям Agile и современной культуре DevOps, нацеленной на сокращение жизненного цикла разработки и непрерывную поставку. Особое внимание уделяется научной новизне современных подходов, таких как DevSecOps, интегрирующий безопасность на каждом этапе разработки, и AIOps, который применяет искусственный интеллект для автоматизации и улучшения ИТ-операций. В заключении подчеркивается, что данная эволюция представляет собой логичный переход от жестких процессов к более гибким, интеллектуальным и безопасным системам разработки.

Annotation. This article examines the evolution of Software Development Life Cycle (SDLC) models, reflecting the industry's pursuit of increased efficiency and quality in product releases. The path is traced from the strict and sequential Waterfall model, through the iterative approach of the Spiral model with its emphasis on risk analysis, to flexible Agile methodologies and the modern culture of DevOps, which aims to shorten the development life cycle and ensure Continuous Delivery. Particular attention is given to the scientific novelty of modern approaches such as DevSecOps, which integrates security at every stage of the development life cycle, and AIOps,

which uses artificial intelligence to automate and improve IT operations. The conclusion emphasizes that this evolution represents a logical transition from rigid processes to more flexible, intelligent, and secure development systems.

Ключевые слова: жизненный цикл разработки ПО (SDLC), Каскадная модель (Waterfall), Спиральная модель, Agile, DevOps, DevSecOps, AIOps, Управление рисками, Непрерывная поставка, Безопасность.

Keywords: Software Development Life Cycle (SDLC), Waterfall model, Spiral model, Agile, DevOps, DevSecOps, AIOps, Risk management, Continuous Delivery, Security.

Разработка программного обеспечения (ПО) – это сложный и многогранный процесс, который со временем претерпел значительные изменения. Для его систематизации и управления были созданы различные модели жизненного цикла разработки (Software Development Life Cycle, SDLC). Важность SDLC заключается в предоставлении структурированного подхода к созданию ПО, что помогает управлять сложностью, ресурсами и рисками. Эволюция этих моделей отражает стремление индустрии к повышению эффективности, скорости и качества выпускаемых продуктов в ответ на растущую сложность проектов и динамичные требования рынка. Этот путь начался со строгой и последовательной каскадной модели, прошел через гибкие итеративные подходы, включая спиральную модель и методологии Agile, и привел к современной культуре DevOps. Однако эволюция на этом не останавливается; появляются новые концепции, такие как DevSecOps и AIOps, которые углубляют интеграцию и интеллектуализацию процессов, формируя научную новизну в подходе к жизненному циклу ПО.

Каскадная модель (Waterfall)

Первой формализованной моделью стала каскадная модель (Waterfall), описанная Уинстоном Ройсом в 1970 году. Интересно, что сам Ройс представлял эту модель как упрощенную и рискованную, предлагая итеративные улучшения, однако именно ее линейная версия получила широкое распространение. Как отмечал Ройс, «разработка крупных программных систем требует более

основательного подхода, и однопроходный последовательный подход несет в себе неотъемлемый риск» [1]. Он также подчеркивал, что в реальном процессе итерации проектирования никогда не ограничиваются последовательным шагом, и модель без итераций рискованна и обречена на провал.

– сбор и анализ требований (Requirements gathering and analysis), то есть полное определение и документирование всех требований к системе до начала разработки. На этом этапе заказчик играет ключевую роль, формулируя свои ожидания;

– проектирование (System Design), а именно разработка архитектуры системы и детальное проектирование компонентов на основе собранных требований;

– разработка (Implementation/Coding), например, написание кода в соответствии с проектной документацией;

– тестирование (Testing) – это проверка разработанного ПО на соответствие требованиям и выявление ошибок. Этот этап начинается только после завершения разработки;

– внедрение (Deployment), например, установка ПО в рабочую среду заказчика;

– поддержка и обслуживание (Maintenance), включает в себя: исправление ошибок, обнаруженных после выпуска, и внесение обновлений.

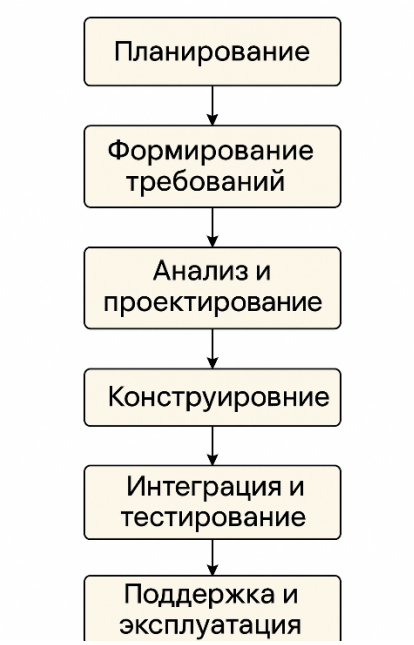

Рисунок 1. Каскадная модель (Waterfall Model)

Переход к следующему этапу возможен только после полного завершения предыдущего. Преимуществами являются четкая структура и простота управления при стабильных требованиях. Однако «жесткость и необходимость точной спецификации требований» [5], а также отсутствие встроенной гибкости в случае изменения требований являются существенными недостатками. Критики отмечают, что тестирование, происходящее в конце цикла разработки, является первым событием, когда проверяются время выполнения, память, передача ввода-вывода и т.д., в отличие от их анализа. Эти явления не поддаются точному анализу. Это приводит к позднему обнаружению ошибок и высокой стоимости их исправления. Кроме того, модель исключает клиента и/или конечного пользователя из большей части процесса, что повышает риск несоответствия продукта ожиданиям.

Спиральная модель

В ответ на недостатки каскадной модели в 1986 году Барри Боэм представил спиральную модель. Этот подход объединил итеративность с системным контролем каскадной модели, сделав особый акцент на анализе рисков. «Отличительной особенностью этой модели является специальное внимание рискам, влияющим на организацию жизненного цикла» [2]. Боэм определил

десять наиболее распространенных рисков, включая дефицит специалистов и нереалистичные сроки.

Жизненный цикл представляет собой спираль, каждый виток которой включает четыре квадранта:

– планирование – определение целей, альтернатив и ограничений;

– анализ рисков– оценка альтернатив, идентификация и разрешение рисков; создание прототипов;

– разработка и тестирование – разработка и тестирование текущей версии продукта.

– оценка – оценка результатов заказчиком и планирование следующей итерации.

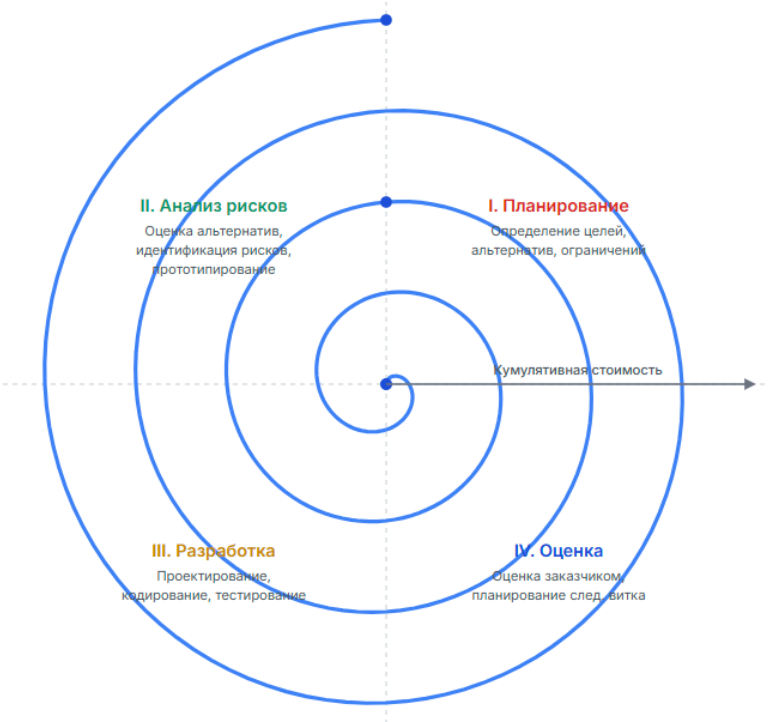

Рисунок 2. Спиральная модель Боэма

Спиральная модель подходит для крупных и сложных проектов с высокими рисками и нечеткими требованиями. Она обеспечивает большую гибкость и раннее обнаружение проблем по сравнению с Waterfall.

Таблица 1

Сравнение Каскадной и Спиральной моделей

Аспект	Каскадная модель	Спиральная модель
Сложность	Простая и понятная	Значительно сложнее
Метод разработки	Последовательный	Эволюционный, итеративный
Управление рисками	Риски выявляются и устраняются после завершения этапов; высокий уровень риска	Риски выявляются и устраняются на ранних стадиях; низкий уровень риска
Гибкость к изменениям	Низкая	Высокая
Подходит для проектов	Малые проекты с четкими требованиями	Крупные, сложные, высокорисковые проекты
Вовлечение заказчика	Минимальное, в основном на начальном этапе	Высокое на протяжении всего процесса
Стоимость	Сравнительно недорогая	Очень дорогая

Гибкие методологии (Agile) как предвестник DevOps

По мере того, как ограничения даже итеративных моделей становились очевидными, в начале 2000-х зародилось движение Agile. Agile – это семейство методологий (Scrum, Kanban и др.), основанных на гибкости, сотрудничестве с заказчиком и инкрементальной поставке работающего ПО. Agile-манифест провозглашает: «Люди и взаимодействие важнее процессов и инструментов» и «Готовность к изменениям важнее следования первоначальному плану» [9].

DevOps: Культура и практика

DevOps – это философия, направленная на изменение культуры взаимодействия между командами разработки (Development) и эксплуатации (Operations). «Совместная работа является основой DevOps. Команды по разработке и эксплуатации образуют функциональную команду, участники которой взаимодействуют, делятся отзывами и совместно работают на протяжении всего цикла разработки и развертывания» [8]. Цель — сократить

жизненный цикл разработки и обеспечить непрерывную поставку (Continuous Delivery).

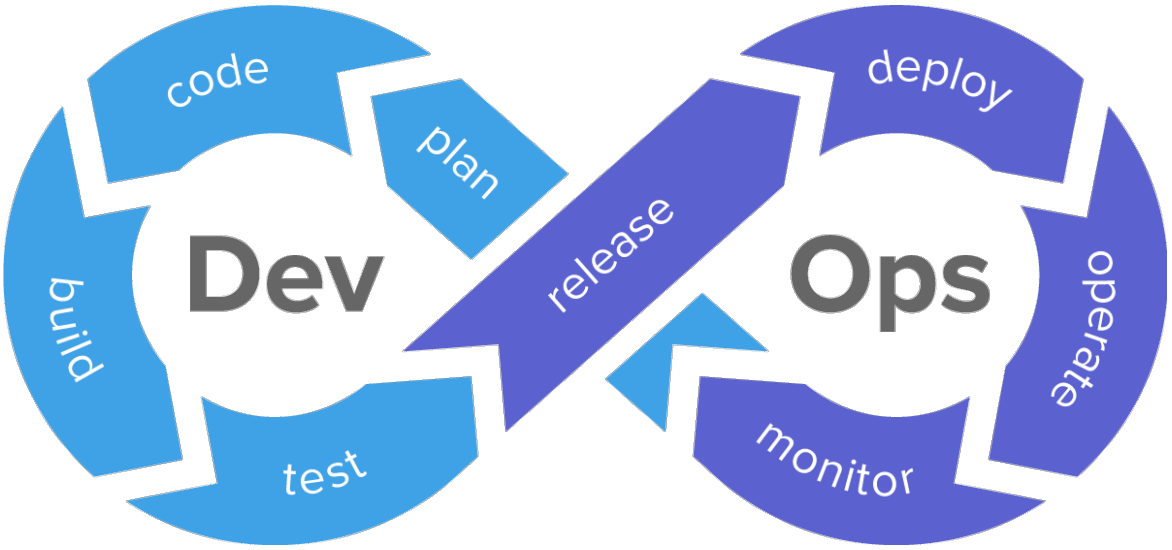

Рисунок 3. Цикл DevOps

Будущее DevOps – DevSecOps и AIOps

Эволюция методологий не останавливается на DevOps. Научная новизна современных подходов заключается в углублении интеграции специализированных функций и интеллектуальных систем непосредственно в жизненный цикл ПО. Два ключевых направления этого развития – DevSecOps и AIOps.

DevSecOps (Development, Security, and Operations) – эта концепция представляет собой расширение DevOps, целью которого является интеграция безопасности на каждом этапе жизненного цикла разработки ПО. Основной принцип – «сдвиг влево» (shifting security left), что означает внедрение практик и инструментов безопасности с самых ранних этапов разработки, а не только на этапе тестирования или после развертывания. «DevSecOps подчеркивает раннюю интеграцию мер безопасности. Это гарантирует выявление и устранение уязвимостей на этапе разработки, а не после развертывания» [8]. Практики

включают автоматизированное тестирование безопасности (SAST, DAST) в конвейерах CI/CD, управление уязвимостями и обеспечение соответствия стандартам. Это не просто добавление безопасности к DevOps, а фундаментальное изменение культуры, где безопасность становится общей ответственностью всей команды.

AIOps (Artificial Intelligence for IT Operations). AIOps использует возможности искусственного интеллекта (ИИ) и машинного обучения (МО) для автоматизации и улучшения ИТ-операций. В условиях экспоненциального роста объемов данных и сложности современных систем, AIOps предлагает интеллектуальные решения для анализа этих данных, выявления аномалий, прогнозирования проблем и автоматического их устранения. Платформы AIOps используют ИИ для автоматического обнаружения аномалий, разрешения инцидентов и рекомендации корректирующих действий без вмешательства человека. Это включает предиктивную аналитику для предотвращения сбоев и оптимизацию распределения ресурсов.

Таблица 2.

Ключевые аспекты DevSecOps и AIOps

Аспект	DevSecOps	AIOps
Основная цель	Интеграция безопасности на всех этапах ЖЦ ПО	Применение ИИ/МО для автоматизации и улучшения ИТ-операций
Ключевой принцип	"Сдвиг безопасности влево"	Предиктивная аналитика, автоматизированное принятие решений и устранение неисправностей
Основные практики	SAST, DAST, автоматизированное управление уязвимостями, безопасность как код	Обнаружение аномалий, анализ первопричин, автоматическое масштабирование,

		интеллектуальное оповещение
Влияние на ЖЦ ПО	Повышение безопасности и соответствия требованиям, снижение рисков	Повышение надежности, доступности и производительности систем, снижение времени простоя
Научная новизна	Внедрение проактивной, непрерывной и автоматизированной безопасности как неотъемлемой части процесса разработки и эксплуатации	Использование ИИ для интеллектуального управления сложными ИТ-системами, переход от реактивного к предиктивному управлению

Заключение

Эволюция от Waterfall к DevOps и далее к DevSecOps и AIOps – это логичный путь от жестких процессов к гибким, непрерывным и интеллектуальным системам. Если Waterfall был эффективен в предсказуемой среде, спиральная модель добавила управление рисками, а Agile – гибкость и ориентацию на клиента. DevOps объединил разработку и эксплуатацию для ускорения поставки. Научная новизна современных подходов, таких как DevSecOps и AIOps, заключается в глубокой интеграции безопасности и искусственного интеллекта, что позволяет не просто быстрее и качественнее разрабатывать ПО, но и создавать более устойчивые, адаптивные и автономные системы, отвечающие вызовам цифровой эпохи. Выбор методологии зависит от множества факторов, но общая тенденция направлена на повышение интеллектуализации и безопасности процессов разработки.

Литература

1. Эволюция методологий разработки: от Waterfall к непрерывной доставке через DevOps // PlaysDev. [Электронный ресурс]. URL: https://playsdev.com/ru/blog/evoljucija-metodologij-razrabotki/ (дата обращения 2025-25-05)

2. Эволюция методологий разработки / Хабр. [Электронный ресурс]. URL: https://habr.com/ru/articles/778502/ (дата обращения: 2025-26-05)

3. 5 факторов, влияющих на выпуски версий программного обеспечения // 4vsar.ru. [Электронный ресурс]. URL: https://www.4vsar.ru/raznoe/5-faktorov-vliyauschih-na-vypyski-versii-programmnogo-obespecheniya.html (дата обращения 2025-28-05)

4. Как выбрать методологию разработки для проекта в 2024 году // Surf. [Электронный ресурс]. URL: https://surf.ru/metodologii-razrabotki-dlya-proekta-2024/ (дата обращения 2025-29-05)

5. Методология разработки Waterfall: как устроена каскадная модель // Timeweb. [Электронный ресурс]. URL: https://timeweb.com/ru/community/articles/metodologiya-razrabotki-waterfall (дата обращения 2025-30-05)

6. Каскадная Модель Жизненного Цикла: Основные Недостатки // LeadStartup. [Электронный ресурс]. URL: https://leadstartup.ru/cascading-model-advantages-disadvantages (дата обращения 2025-04-06)

7. Spiral Model Спиральная модель и архитектура разработки программного обеспечения // Janberg [Электронный ресурс]. URL: https://janberg.by/spiral-model-spiralnaya-model-i-arxitektura-razrabotki-programmnogo-obespecheniya/ (дата обращения 2025-02-06)

8. Введение в DevOps: Ключевые принципы и практики // Serverspace. [Электронный ресурс]. URL: https://serverspace.ru/about/blog/vvedenie-v-devops-klyuchevye-princzipy-i-praktiki/ (дата обращения 2025-06-06)

9. Что такое Agile на самом деле? // Agile Consulting [Электронный ресурс]. URL: https://onagile.ru/trends/agile/what-is-agile (дата обращения 2025-05-06)