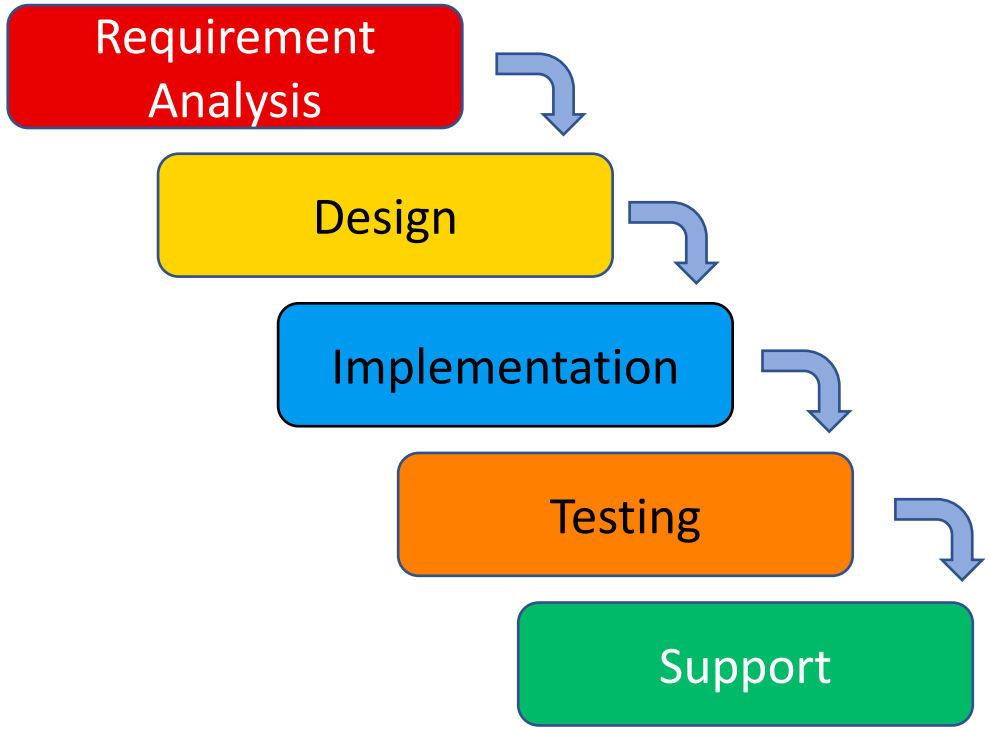
Применение инструментов автоматического анализа программ в цикле разработки безопасного ПО

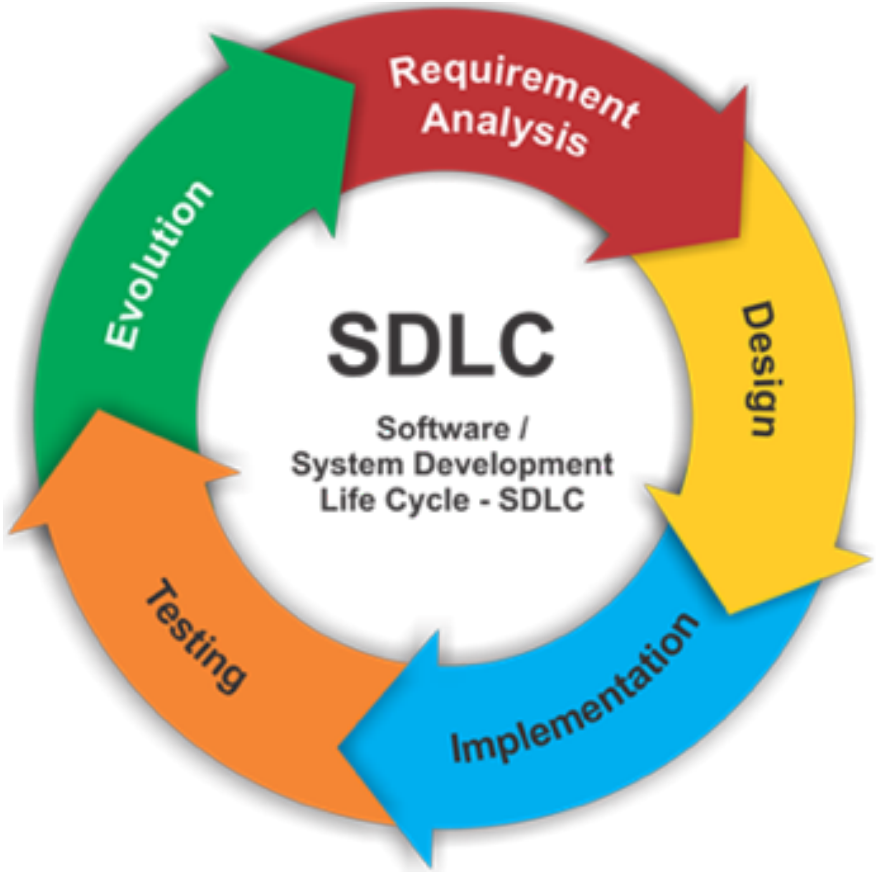
OS DAY

Москва, 2019

Герасимов А. Ю.
Старший научный сотрудник
ИСП РАН

Цикл разработки ПО





Цикл разработки безопасного ПО

- ГОСТ Р 56939-2016 «Разработка безопасного программного обеспечения. Общие требования.»
- ГОСТ Р ИСО/МЭК 12207-2010 «Информационная технология, Системная и программная инженерия. Процессы жизненного цикла программных средств.»
- ГОСТ Р ИСО/МЭК 15408 «Методы и средства обеспечения безопасности. Критерии оценки безопасности информационных систем.»

- OWASP, NASA Security Guidelines, другие.

- Методика выявления уязвимостей и недекларированных возможностей в программном обеспечении. ФСТЭК России. 2019

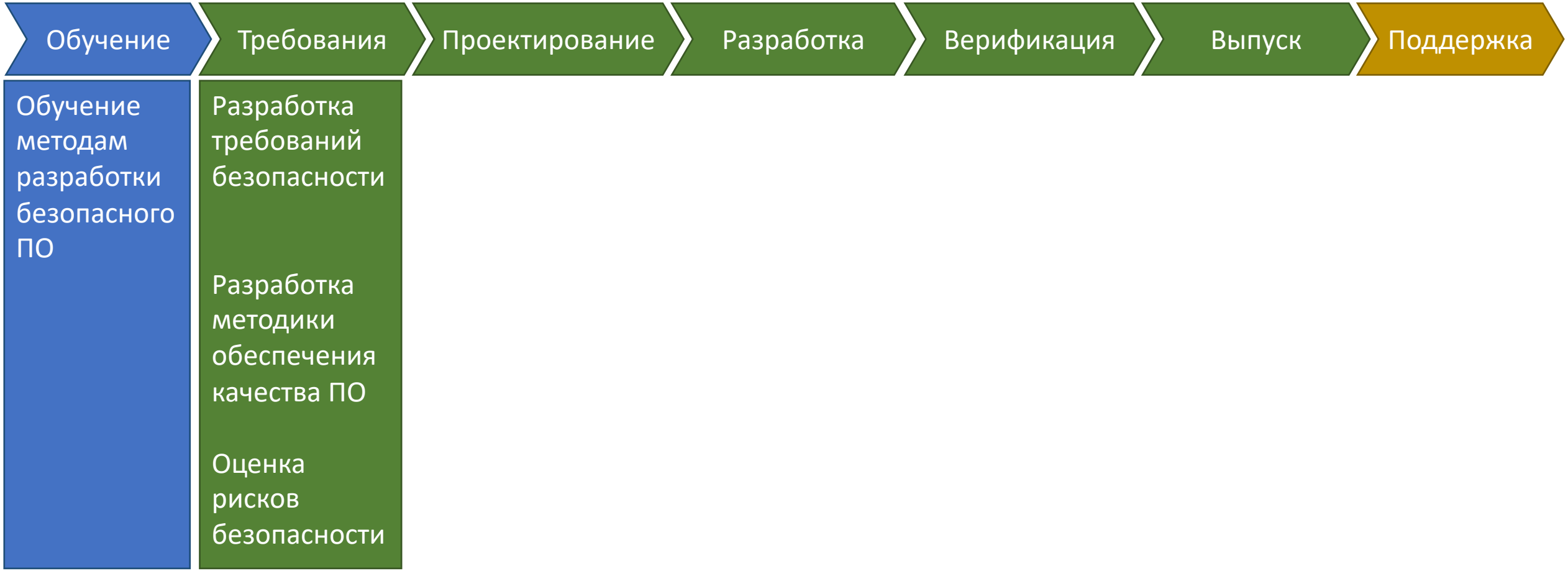
Важность обучения

Обучение методам разработки безопасного ПО

Перечень знаний, необходимый для создания безопасного ПО:

- Виды угроз, уязвимостей и ошибок в программах
- Методы предотвращения появления уязвимостей и ошибок в программах
- Методы обнаружения уязвимостей и ошибок в программах
- Перечень и возможности инструментов поддержки процесса разработки ПО
- Методики и методологии разработки безопасного ПО

Цикл разработки безопасного ПО



Цикл разработки безопасного ПО

Цикл разработки безопасного ПО

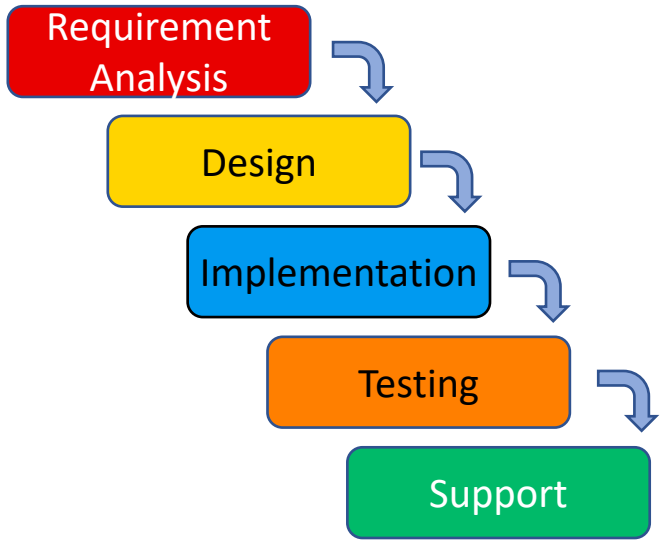
Цикл разработки безопасного ПО

Цикл разработки безопасного ПО

Цикл разработки безопасного ПО

Методология

- Применение методик обеспечения безопасности ПО независимо от применяемой методологии разработки
- Непрерывное внимание к безопасности ПО начиная с ранних стадий жизненного цикла
- Применение инструментов анализа безопасности ПО на всех стадиях жизненного цикла
- Непрерывное повышение квалификации команды в области безопасности ПО



Инструментальная поддержка

Статический анализ кода программ:

1. Инспекция кода
2. Линтеры (lint)
3. Предупреждения компилятора
4. Инструменты статического анализа программ.

Доступные на рынке инструменты:

- Svace (ИСП РАН, Россия)
- Klocwork (RogueWave, США)
- Coverity (Synopsys, США)
- Java FindBugs (Maryland University, США)
- Pyhton: PyLint, PyType
- JavaScript: JSHint, JSLint, Google Closure Linter
- …

Характеристики:

- Поддерживаемые языки программирования
- Обнаруживаемые ошибки
- Поддерживаемые уровни анализа
- Производительность
- Интеграция с IDE и CI

Применение:

- Выбор инструмента
- Настройка
- Анализ и разметка предупреждений
- Интеграция с CI

Инструментальная поддержка

Динамический анализ кода программ:

1. Фаззинг + Sanitizers
2. Динамическое символьное исполнение
3. Комбинированные инструменты.

Доступные на рынке инструменты:

- Anxiety (ИСП РАН, Россия)
- ИСП Фаззер (ИСП РАН, Россия)
- angr (Open source)
- Americal Fuzzy Lop (Open source)
- Driller (Open source)
- Valgrind:memcheck, TaintGrind,… (Open source)
- …

Характеристики:

- Поддерживаемые платформы
- Обнаруживаемые ошибки
- Поддерживаемые методы анализа
- Производительность

Применение:

- Выбор инструмента
- Настройка
- Интеграция с CI

Инструментальная поддержка

Сопровождение разработанного ПО:

1. Поиск колонов в машинном коде
2. Статический анализ машинного кода

Доступные на рынке инструменты:

- Code Clone Detector (ИСП РАН, Россия)
- BinSIDE (ИСП РАН, Россия)
- BitBlaze/vine (Berkley University, США)
- CodeSurfer/x86 (GammaTech, США)

Характеристики:

- Поддерживаемые платформы
- Обнаруживаемые ошибки
- Поддерживаемые методы анализа
- Производительность

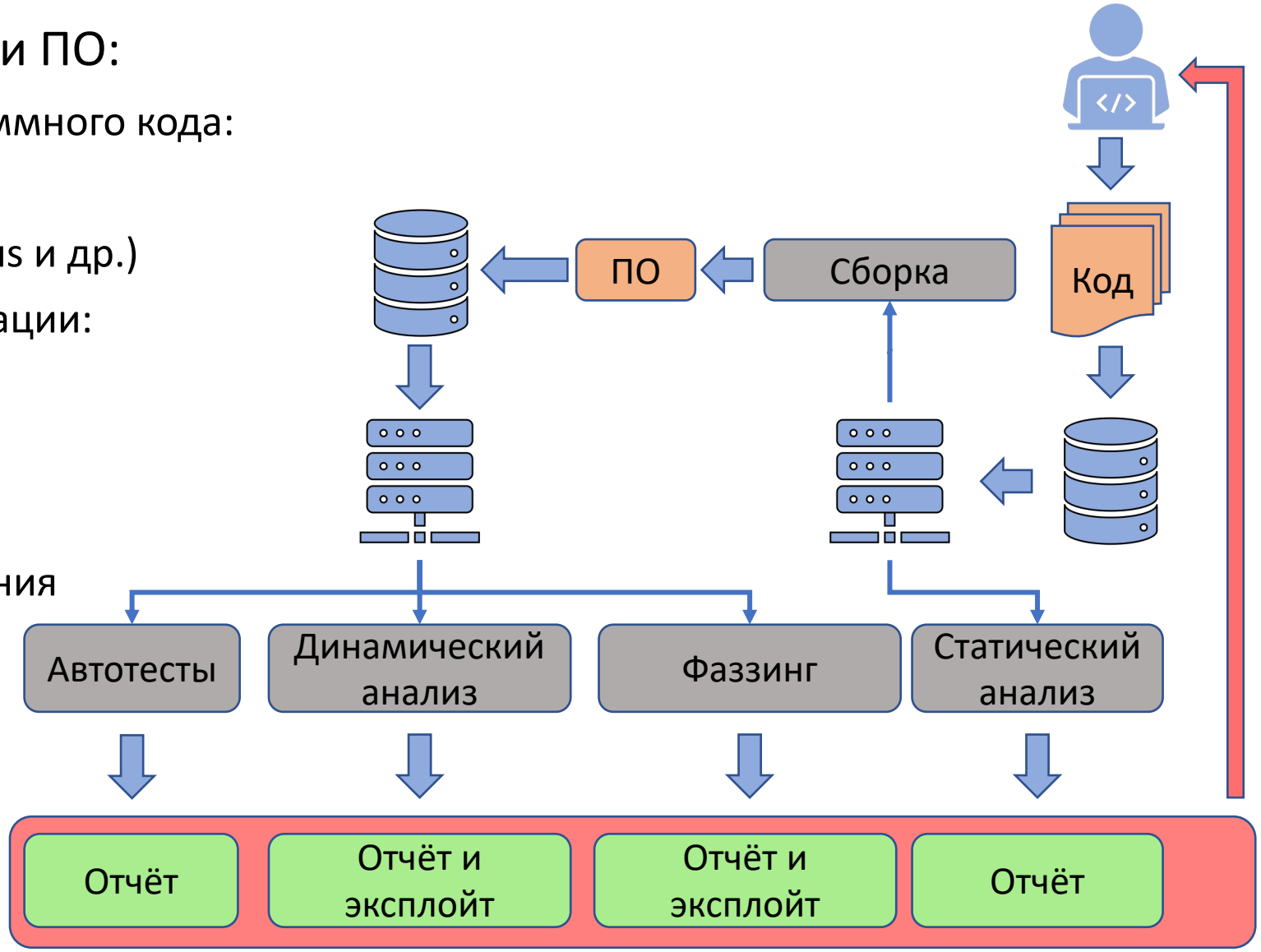
Применение:

- Выбор инструмента
- Настройка
- Интеграция с CI

Автоматизация процесса

Технологический цикл разработки ПО:

1. Системы версионирования программного кода:
   - Исходный код (CVS, Git и др.)
   - Бинарный код (Artifactory, Nexus и др.)
2. Инструменты непрерывной интеграции:
   - TeamCity, Jenkins и др.
3. Компиляторы
4. Статический анализ
5. Системы автоматизации тестирования
6. Фаззинг
7. Динамический анализ



Спасибо за внимание.