Software Engineering Conference Russia 2018

October 12-13
Moscow

Подводные камни Security Development Lifecycle

Болдырев А.Г.

Компания «Код Безопасности»

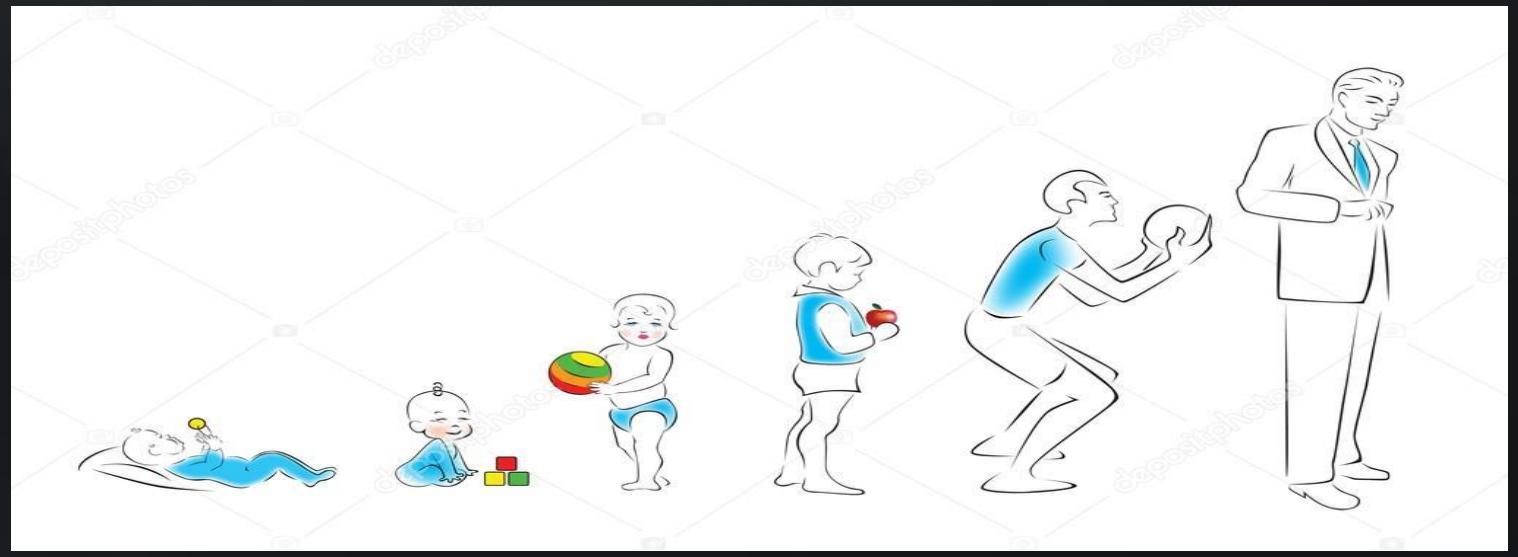
Цель доклада:

- знакомство с методологией SDL
- обзор практики внедрения статического анализа

Я - исследователь безопасности ПО

1. Опыт работы в области ИБ более 10 лет
2. Аудит безопасности собственных продуктов
3. Исследование защищенности информационных систем
4. Email: boldyrev_pnz@mail.ru

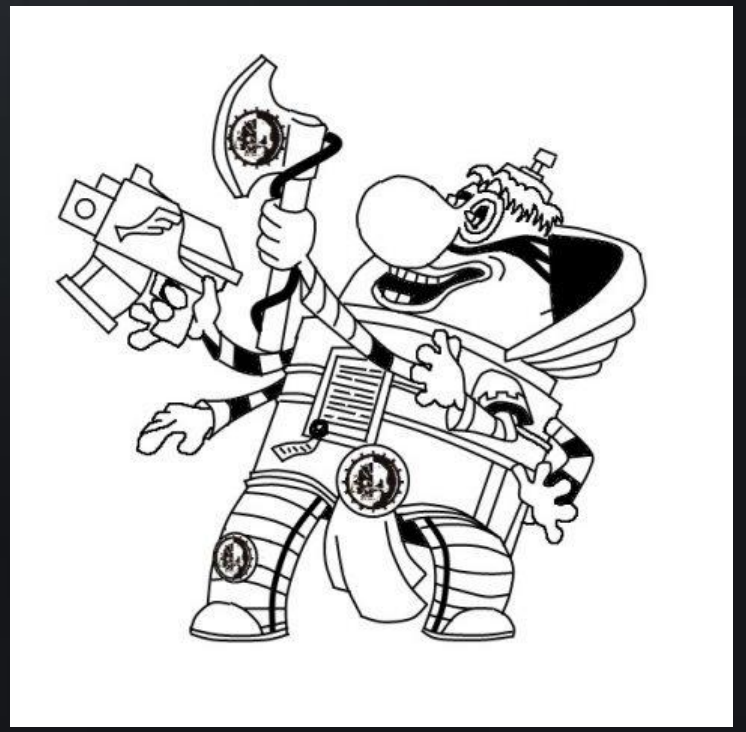
Жизненный цикл разработки программного обеспечения



Результат

Ожидания

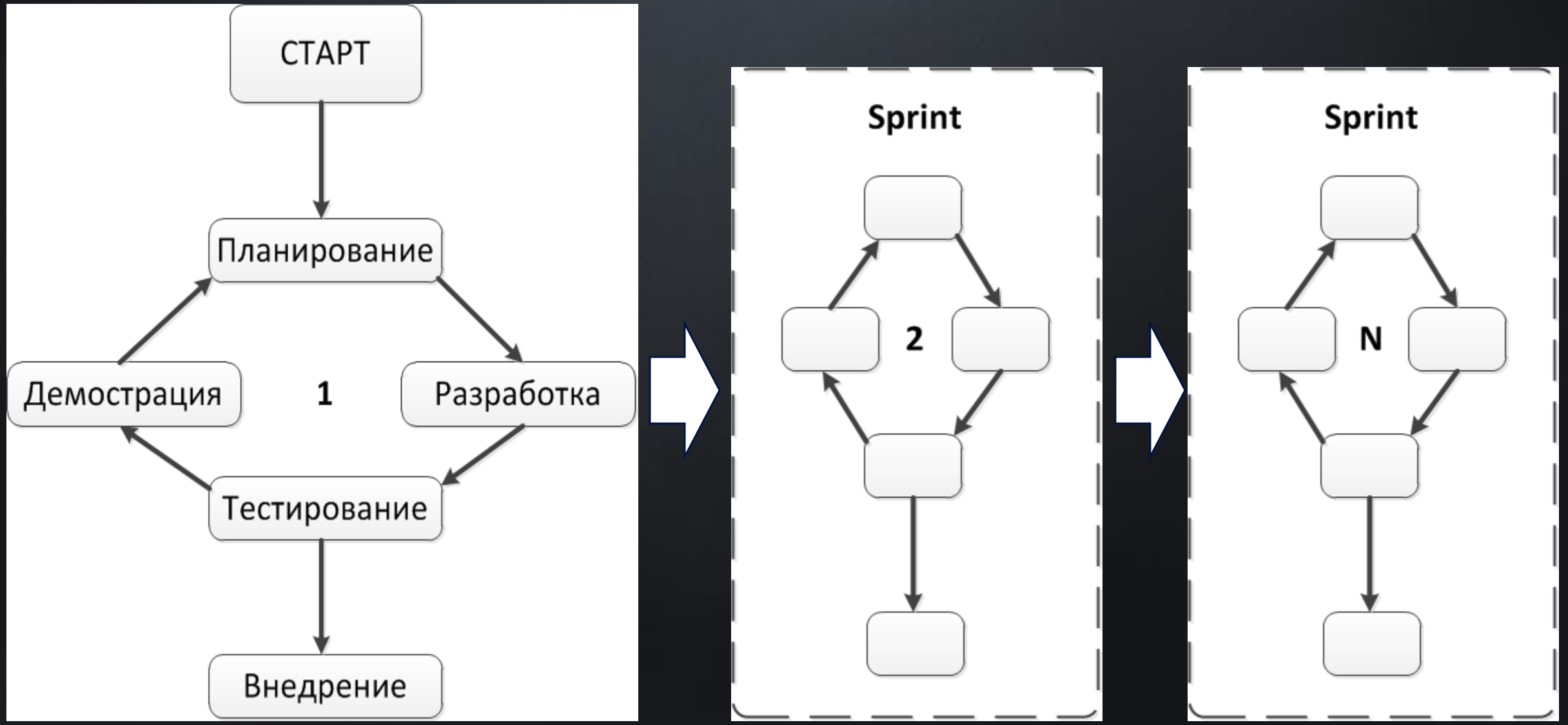
Реальность



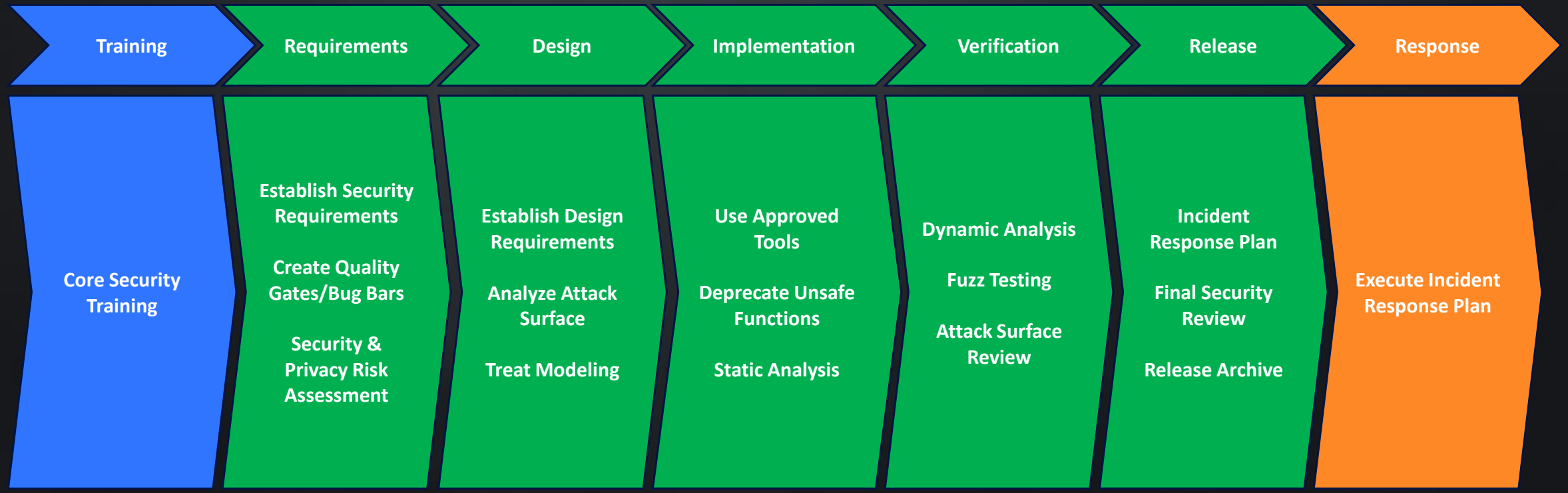
Waterfall


1
Проектирование
2
Дизайн
3
Кодирование
4
Тестирование
5
Поддержка

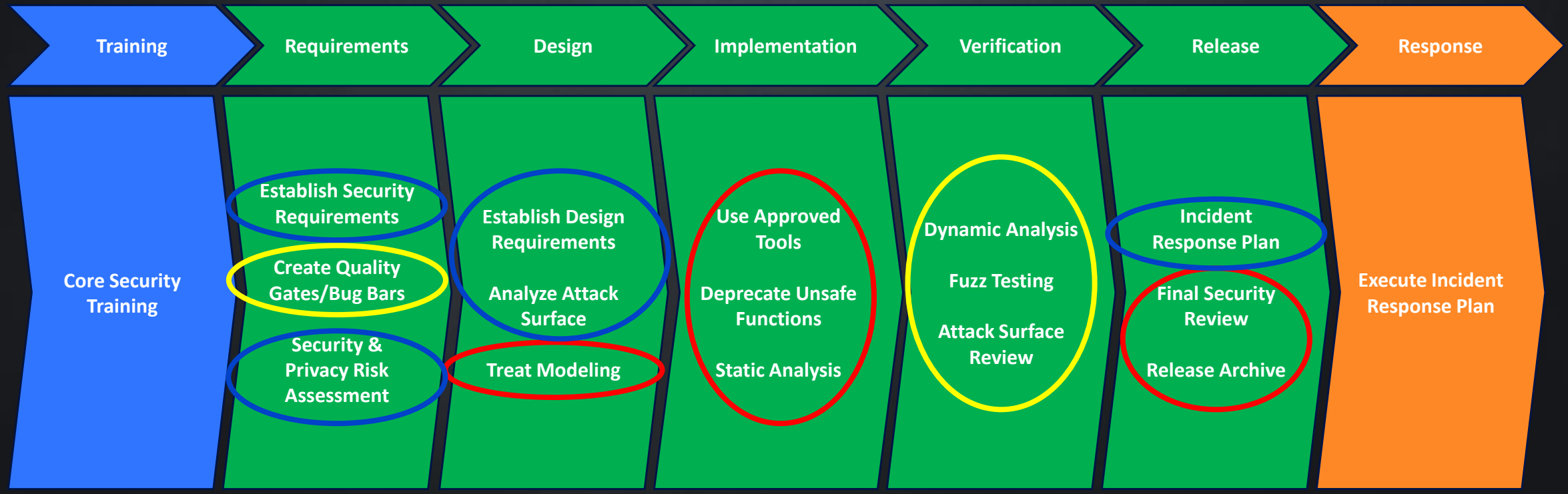
Agile


СТАРТ
Планирование
Демострация
1
Разработка
Тестирование
Внедрение
Sprint
2
Sprint
N

SDL for waterfall



SDL for agile

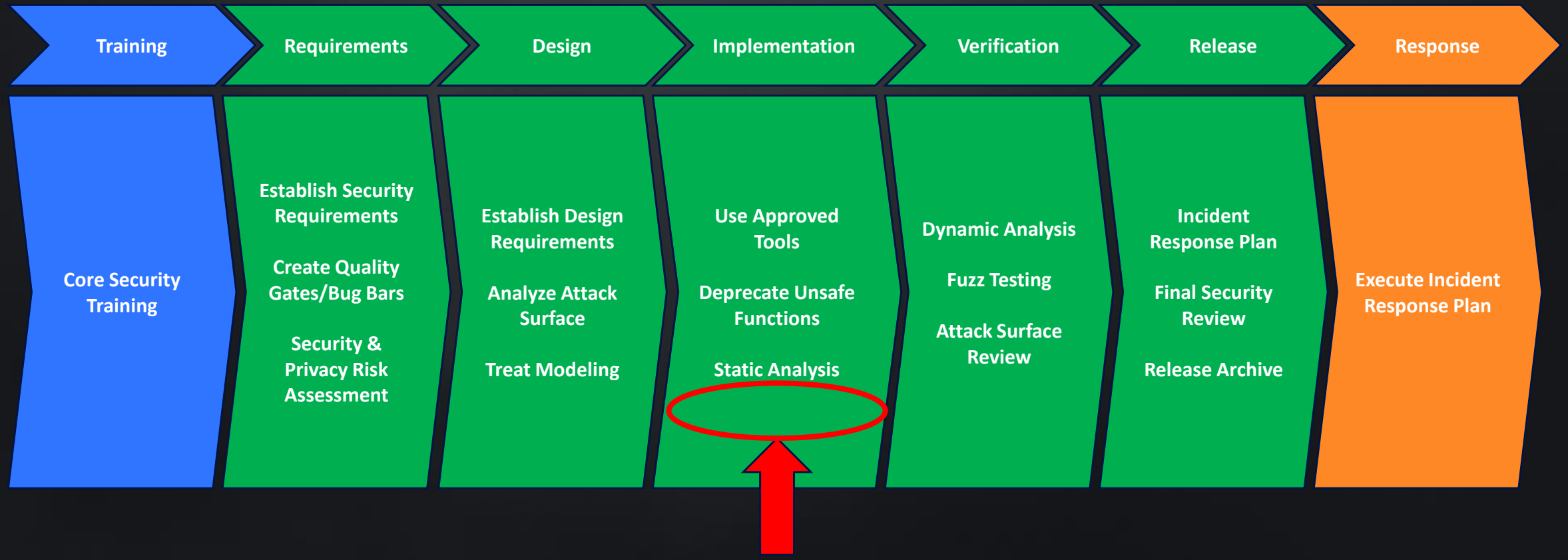


EVERY-SPRINT	Каждый спринт	
BUCKET	Несколько спринтов	
ONE-TIME	Однажды при старте	

SDL vs ГОСТ

Процесс	Фаза SDL	ГОСТ Р 56939
Обучение сотрудников	Training	Меры при менеджменте людских ресурсов
Определение требования безопасности	Requirements	Меры при анализе требований ПО
Моделирование угроз	Design	Меры при проектировании архитектуры
Доверенные средства разработки	Implementation	Меры при менеджменте инфраструктуры среды разработки
Статический анализ исходных текстов ПО	Implementation	Меры при конструировании ПО
Динамический анализ	Verification	Меры при выполнении квалификационного тестирования
Тестирование на проникновение	Verification	Меры при выполнении квалификационного тестирования
Фаззинг-тестирование	Verification	Меры при выполнении квалификационного тестирования
Реагирование на инцидент	Release, Response	Меры при решении проблем ПО в процессе эксплуатации

Статический анализ в SDL



Статический анализ

Статический анализ исходных текстов ПО

Этапы внедрения:

- Составление перечня языков программирования
- Выбор статического анализатора
- Внедрение статического анализа
- Анализ и обработка результатов

Статический анализ исходных текстов ПО

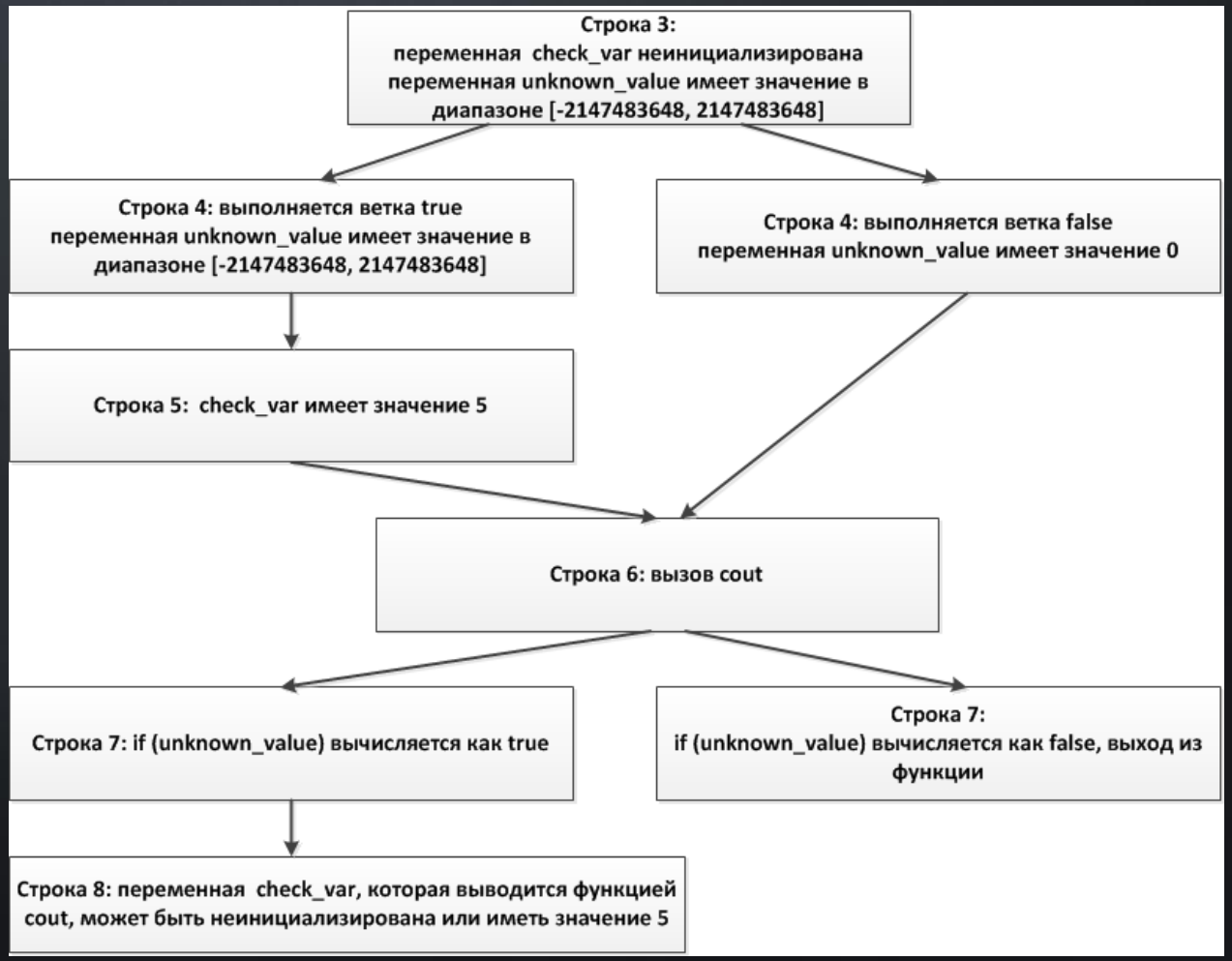
```
1: #include <iostream>
2: void test_func(int unknown_value){
3:    int check_var;
4:    if (unknown_value)
5:           {check_var = 5;}
6:    std::cout<<"hi"<<std::endl;
7:    if (!unknown_value)
8:           {std::cout<<check_var;}//если unknown_value=0
9: }
```

Граф достижимых состояний

```
1: #include <iostream>
2: void test_func(int unknown_value){
3:       int check_var;
4:       if (unknown_value)
5:                {check_var = 5;}
6:       std::cout<<"hi"<<std::endl;
7:       if (!unknown_value)
8:                {std::cout<<check_va
9: }
```

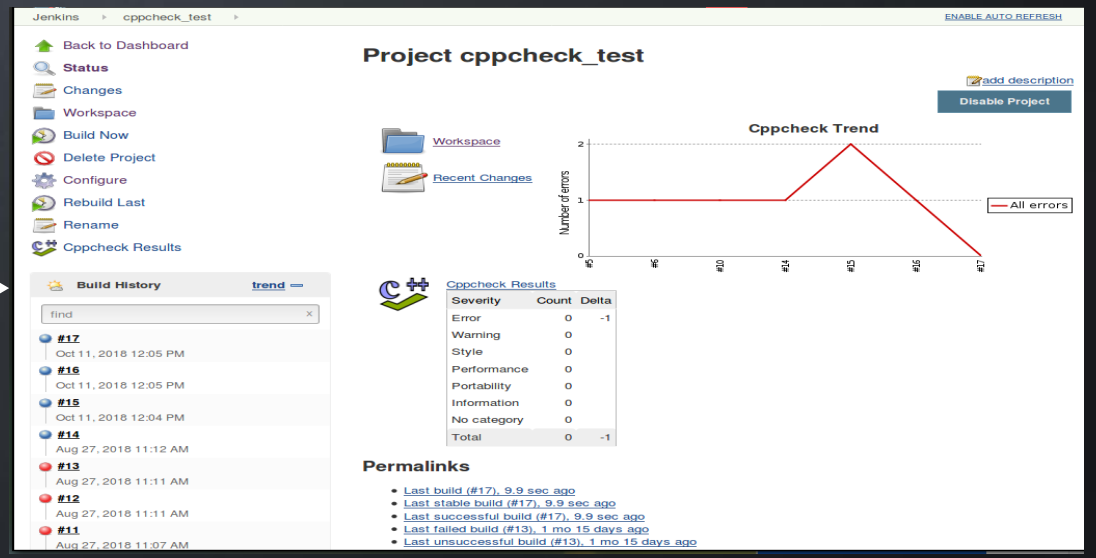
Граф потока выполнения

```
1: #include <iostream>
2: void test_func(int unknown_value){
3:       int check_var;
4:       if (unknown_value)
5:                {check_var = 5;}
6:       std::cout<<"hi"<<std::endl;
7:       if (!unknown_value)
8:                {std::cout<<check_var;}
9: }
```

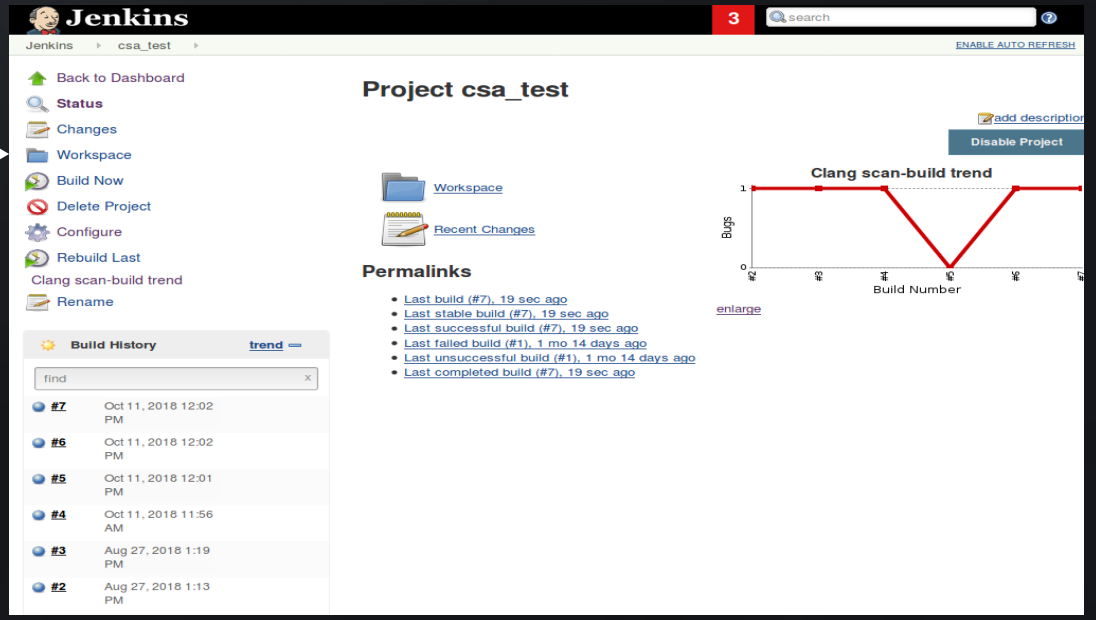


Внедрение статического анализа

Cppcheck plugin



Scan-build plugin



Проблемы внедрения статического анализа и пути решения

Этап	Проблема	Решение
Составление перечня языков программирования	Нет	
Выбор статического анализатора	Большой выбор, разные методы	Использование нескольких технологий анализа, апробирование на тестовых примерах
Внедрение статического анализа	Знание условий сборки, ограничение реализации	Тесное сотрудничество с командой разработки
Анализ и обработка результатов	Большое количество “ложных” срабатываний	Тонкая настройка статического анализатора под конкретный проект, введение процесса автоматической отбраковки “ложных” срабатываний

Выводы

- Выполнили обзор основных подходов к разработке ПО (waterfall и agile)
- Рассмотрели фазы методологии SDL
- Убедились, что ГОСТ Р 56939-2016 не противоречит SDL
- Выделили проблемы и пути их решения при внедрении статического анализа, как части SDL

Спасибо за внимание!