SDL, надежность и инженерная культура

Александр Дубинин
Группа компаний YADRO

О группе компаний ЯДРО

YADRO — группа российских технологических компаний, объединяющая направления разработки и производства вычислительных платформ, систем обработки и хранения данных, телекоммуникационного и сетевого оборудования, персональных и «умных» устройств, микропроцессорных ядер и fabless-разработку микропроцессоров.

За последние несколько лет численность сотрудников увеличилась почти в 10 раз и теперь более 2 500 человек, из которых существенную часть сотрудников (80%) составляют инженеры. Соответственно, возросла и сложность процессов, как управленческих, так и производственных.

О группе компаний ЯДРО: yadro.com/ru/company

Продукты ЯДРО: yadro.com/ru/products

Продвижение инженерного образования и культуры

Выпускникам технических ВУЗов нередко не хватает базовых знаний, научной и инженерной культуры. Что мы делаем для улучшения ситуации?

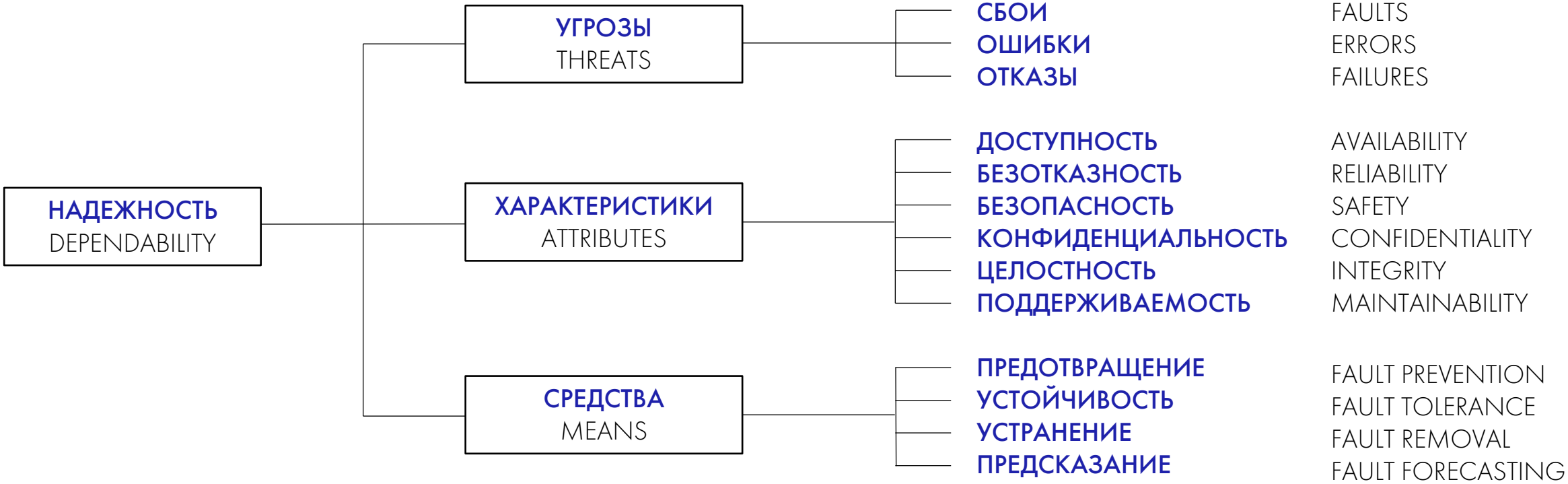
- Организован Портал Инженерной Культуры — **«Истовый Инженер»** (engineer.yadro.com) с общедоступными тренингами и лекциями
- Наши эксперты выступают с докладами на тематических конференциях
- Ведем работу с ВУЗами, читаем курсы лекций (МФТИ, ИТМО, СПБГУ и другие)
- Проводим занятия и олимпиады для школьников (летний лагерь «Сириус»)
- Берем на работу студентов старших курсов
- Постоянно проводим внутренние тренинги и лекции

SDL(SSDLC) как процесс повышения надежности

SDL = SSDLC = Secure System/Software Development Lifecycle

В английском языке синоним слова Security — Dependability или «Надежность»

Надежность компьютерной системы — способность выполнять функцию (или обеспечивать сервис — т.е. видимое пользователем поведение), результатам которой можно в разумных пределах доверять.



Algirdas Avižienis, Jean-Claude Laprie, Brian Randell. «Fundamental Concepts of Dependability»

Основные критерии надежности

AVAILABILITY **ДОСТУПНОСТЬ**	Готовность системы предоставлять сервис, выполнять функцию
RELIABILITY **БЕЗОТКАЗНОСТЬ**	Надежность в смысле достоверности результатов работы, способность давать корректный результат в течении длительного времени
SAFETY **БЕЗОПАСНОСТЬ**	Отсутствие катастрофических последствий для пользователя или окружения
CONFIDENTIALITY **КОНФИДЕНЦИАЛЬНОСТЬ**	Отсутствие неавторизованного доступа к информации
INTEGRITY **ЦЕЛОСТНОСТЬ**	Отсутствие некорректных изменений состояния системы
MAINTAINABILITY **ПОДДЕРЖИВАЕМОСТЬ**	Способность к восстановлению и модификации
ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ	Конфиденциальность + целостность + доступность
FAULT **НЕИСПРАВНОСТЬ**	Переход из состояния корректного выполнения функции в состояние некорректного выполнения функции
ERROR **ОШИБКА**	Следствие неисправности, часть состояния системы, при котором проявляется сбой
FAILURE **СБОЙ**	Ситуация, когда ошибка приводит к существенному изменению функционирования системы в состояние, не соответствующему спецификации

Надежность программных систем и SDL

Для программных систем также можно определить:

CORRECTNESS **КОРРЕКТНОСТЬ**	Способность системы функционировать согласно спецификации, при условии использования в ее рамках
ROBUSTNESS **ЖИВУЧЕСТЬ**	Способность системы не приносить вреда в случае ошибочного выхода условий работы за рамки спецификации
SECURITY **БЕЗОПАСНОСТЬ**	Способность системы не приносить вреда в случае злонамеренного выхода условий работы за рамки спецификации

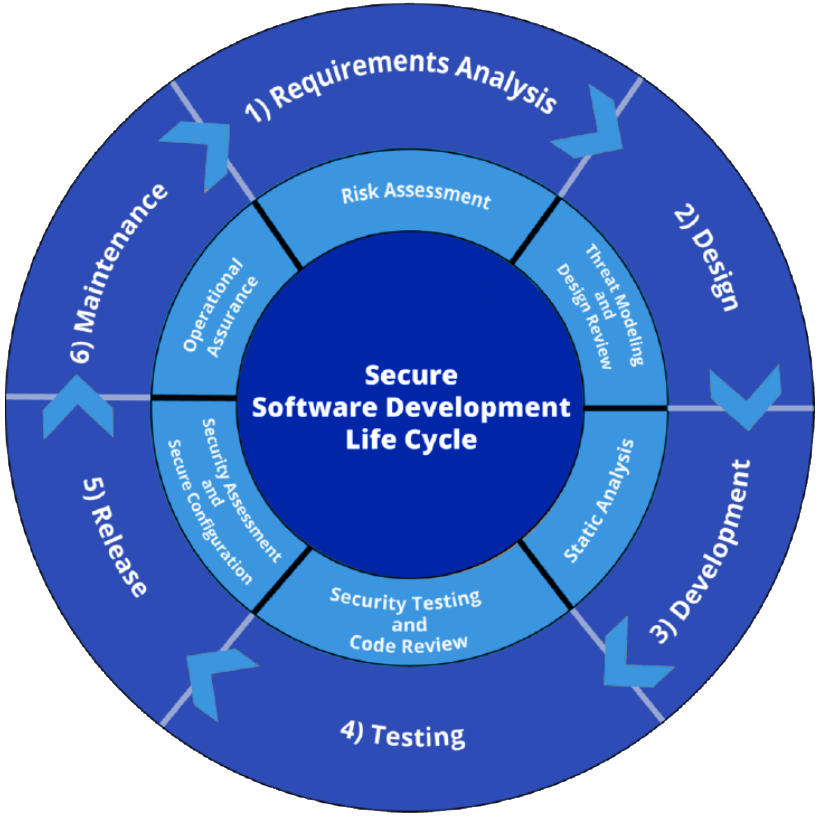
SDL(SSDLC) работает на улучшение этих характеристик

SDL и жизненный цикл продукта (PLC)

SDL (SSDLC, процесс РБПО) — дополнительный слой поверх PLC (Product Life Cycle) или Жизненного Цикла Продукта.
И если для продукта не выстроен жизненный цикл, SDL опираться не на что...

Часто под SDL подразумевают тестирование, инструменты и артефакты процесса разработки... Но на стадии проектирования тоже есть критичные для SDL этапы жизненного цикла (PLC):

- Создание концепции продукта и определение сценариев использования
- Формирование функциональных и нефункциональных требований
- Создание архитектуры продукта



Инженерные принципы проектирования (1)

- Конструктивизм, где «функция определяет форму», а назначение и функциональность продукта определяют его архитектуру
- Функциональный минимализм — в состав системы входят только нужные для выполнения требуемых функций компоненты, правило «Бритвы Оккама»
- K.I.S.S. — Keep It Simple Stupid или «Чем проще тем надежнее» — применяем простые и понятные решения

Обоснование из теории надежности:

Вероятность безотказной работы – критерий, который достаточно полно характеризует надежность системы и может быть определен на стадии проектирования по вероятности безотказной работы составляющих ее элементов(компонентов)

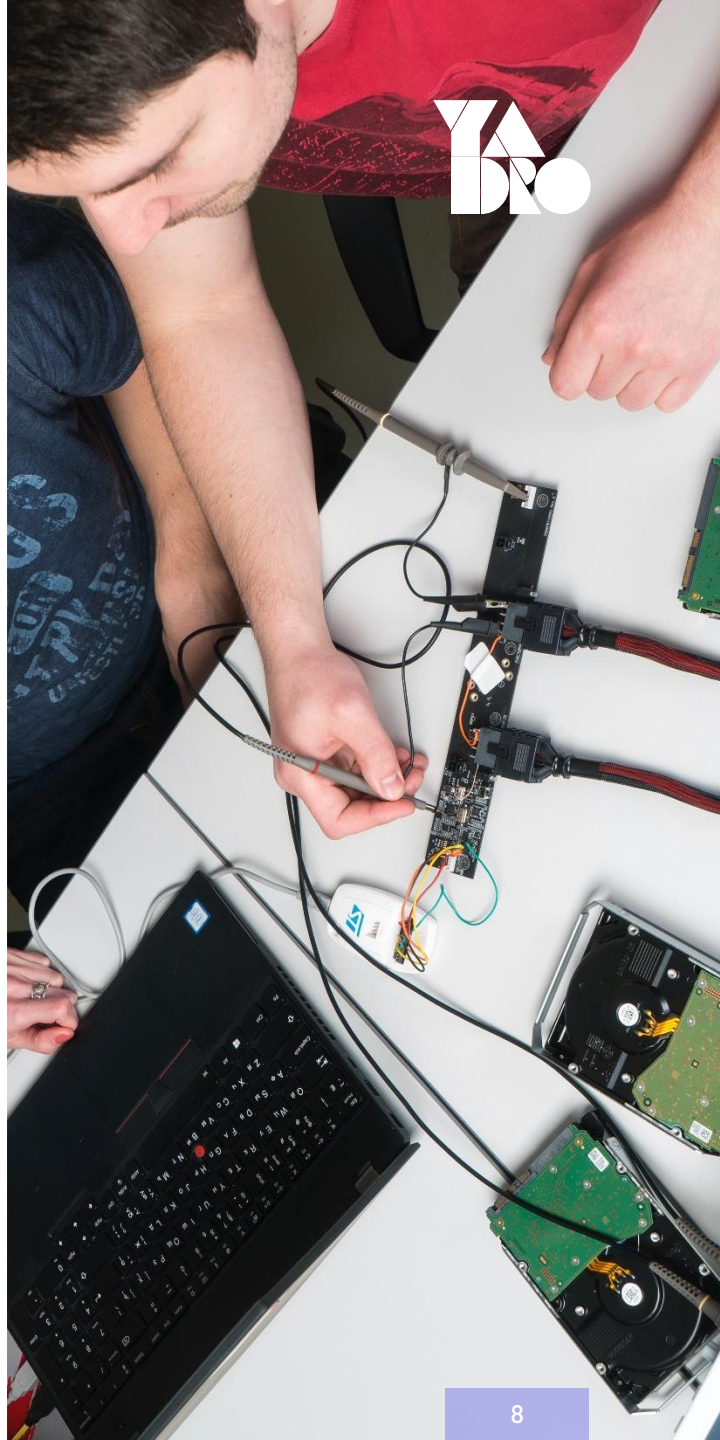
Вероятность безотказной работы системы взаимосвязанных **элементов** – определяется как произведение вероятностей безотказной работы каждого элемента в этой группе:

$$P(t) = P_1(t) \cdot P_2(t) \cdot \ldots \cdot P_n(t) = \prod_{k=1}^{n} P_k(t)$$

Определение вероятности безотказной работы:

Для физических компонентов — наработка на отказ

Для программных компонентов — динамический анализ



Инженерные принципы проектирования (2)

Практические правила при создании сложной системы:

- Не используем «всемогуторы»
- Применяем правильный инструмент для решения правильной задачи
- Предпочтение отдается атомарным, слабо связанным компонентам
- Полное владение **всеми** компонентами продукта (технологической цепочкой)

Владение компонентами подразумевает наличие возможности и экспертизы для их модификации, доработки и полноценного тестирования.

Использование Open Source не предоставляет такой возможности для **всех** из них, не определяет ответственность в случае возникновения проблем.

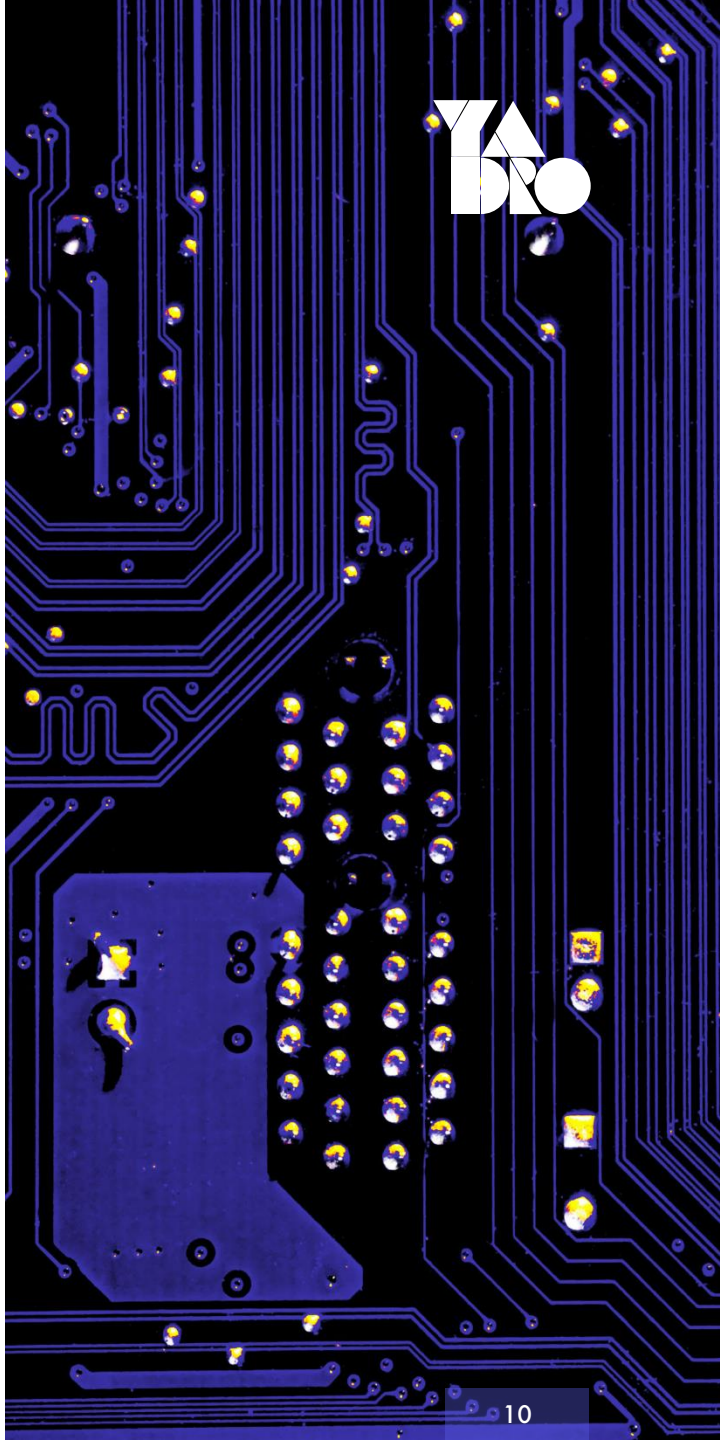
Архитектура!

Если архитектура не задокументирована — то ее нет!

- ТЗ, ПЗ и эскизный проект (PRD, HLA)
- Технический проект (HLD)

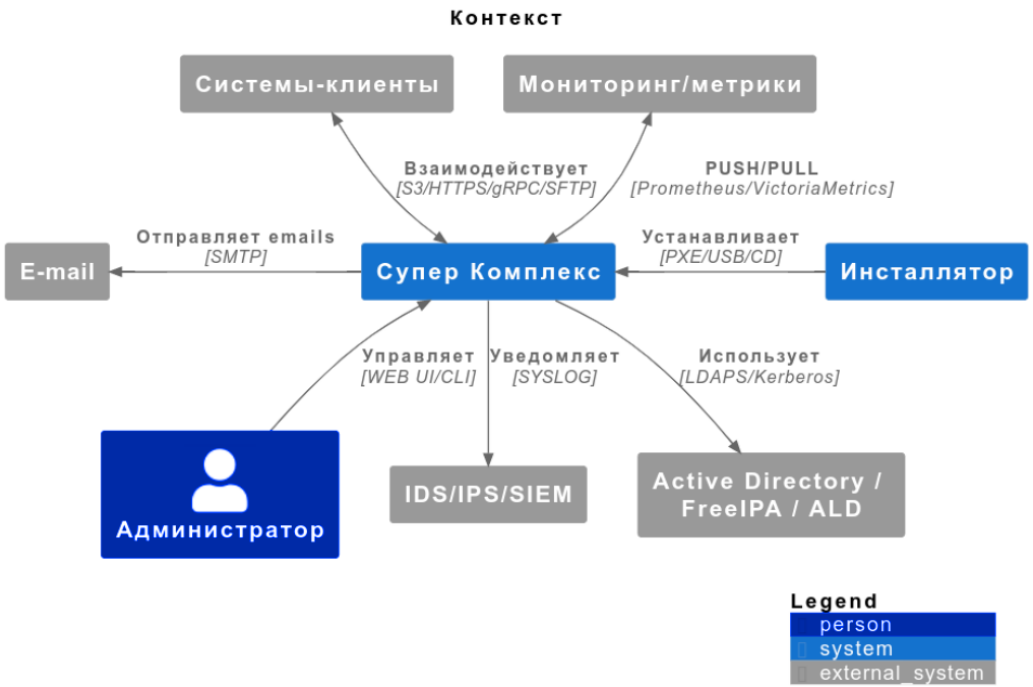
Архитектура должна быть понятна всем

- Модель C4 (K4): Контекст, Контейнеры (подсистемы), Компоненты, Код
- Прослеживаемость от «хотелок» (user stories) до реализации и обратно
- Журнал Архитектурных Решений (Architectural Decision Records Log)
- «Фигак-фигак продакшен» и как его избежать



Документация — это просто

- Система «Единого источника»
- DITA, DocBook, и прочее — документация как код
- Диаграммы как код
- Сквозное версионирование
- Автоматическая генерация документации




```
@startuml Context
!include <c4/C4_Context.puml>
title Контекст

Person(admin, "Администратор")
System(node, "Супер Комплекс")
System(installer, "Инсталлятор")

System_Ext(consumer, "Системы-клиенты")
System_Ext(mail_system, "E-mail")
System_Ext(ids, "IDS/IPS/SIEM")
System_Ext(ldap, "Active Directory / FreeIPA / ALD")
System_Ext(monitor, "Мониторинг/метрики")

Rel_U(admin, node, "Управляет", "WEB UI/CLI")
BiRel(consumer, node, "Взаимодействует", "S3/HTTPS/gRPC/SFTP")
Rel_L(node, mail_system, "Отправляет emails", "SMTP")
Rel(node, ids, "Уведомляет", "SYSLOG")
Rel(node, ldap, "Использует", "LDAPS/Kerberos")
BiRel(monitor, node, "PUSH/PULL", "Prometheus/VictoriaMetrics")
Rel_L(installer, node, "Устанавливает", "PXE/USB/CD")

SHOW_LEGEND()
@enduml
```

Немного про инструменты SDL

Взаимосвязь компонентов:

- Анализ **архитектуры**
- Анализ потоков информации

Знаем про каждый байт - откуда он взялся, почему и зачем нам нужен:

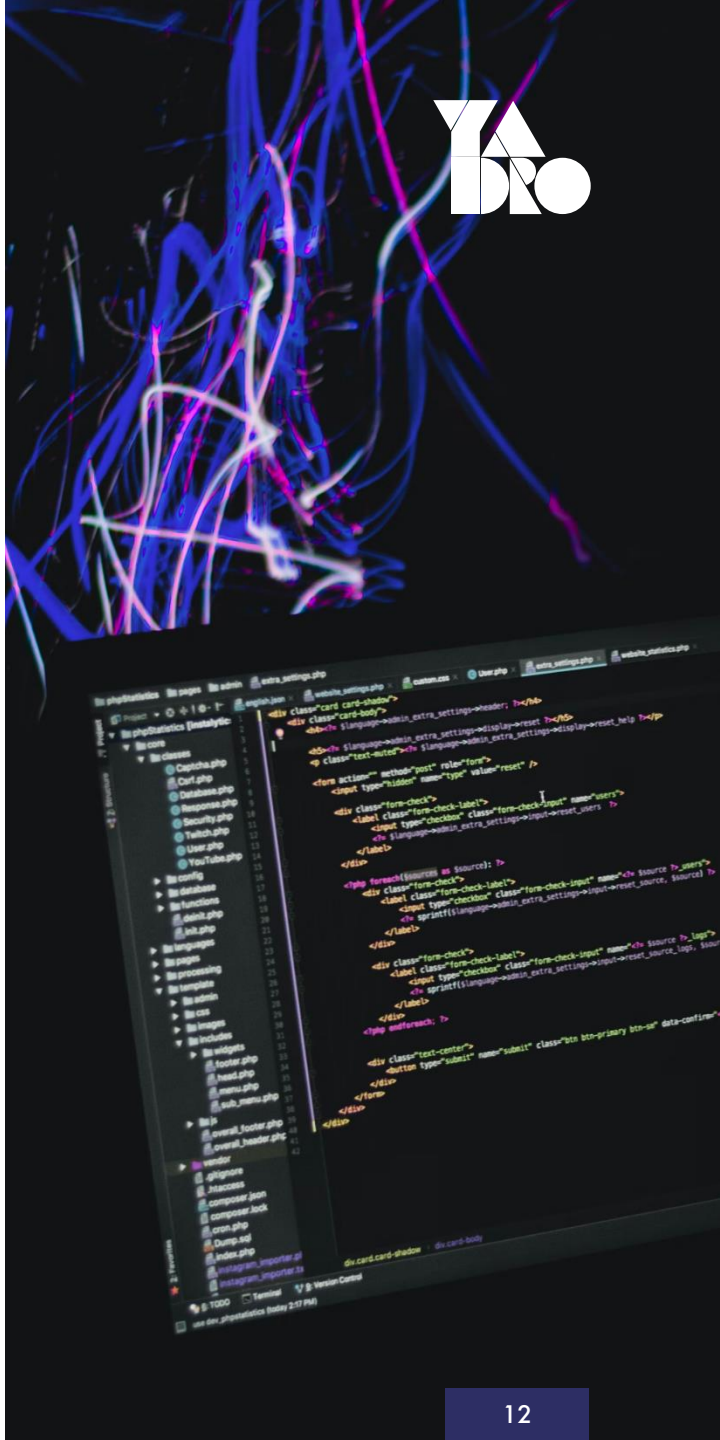
- Анализ SBOM (Dependency Track, Code Scoring)

Качество кода:

- Линтеры
- Статический анализ

Надежность компонентов — динамический анализ:

- Модульное тестирование
- Фаззинг



Внедрение SDL

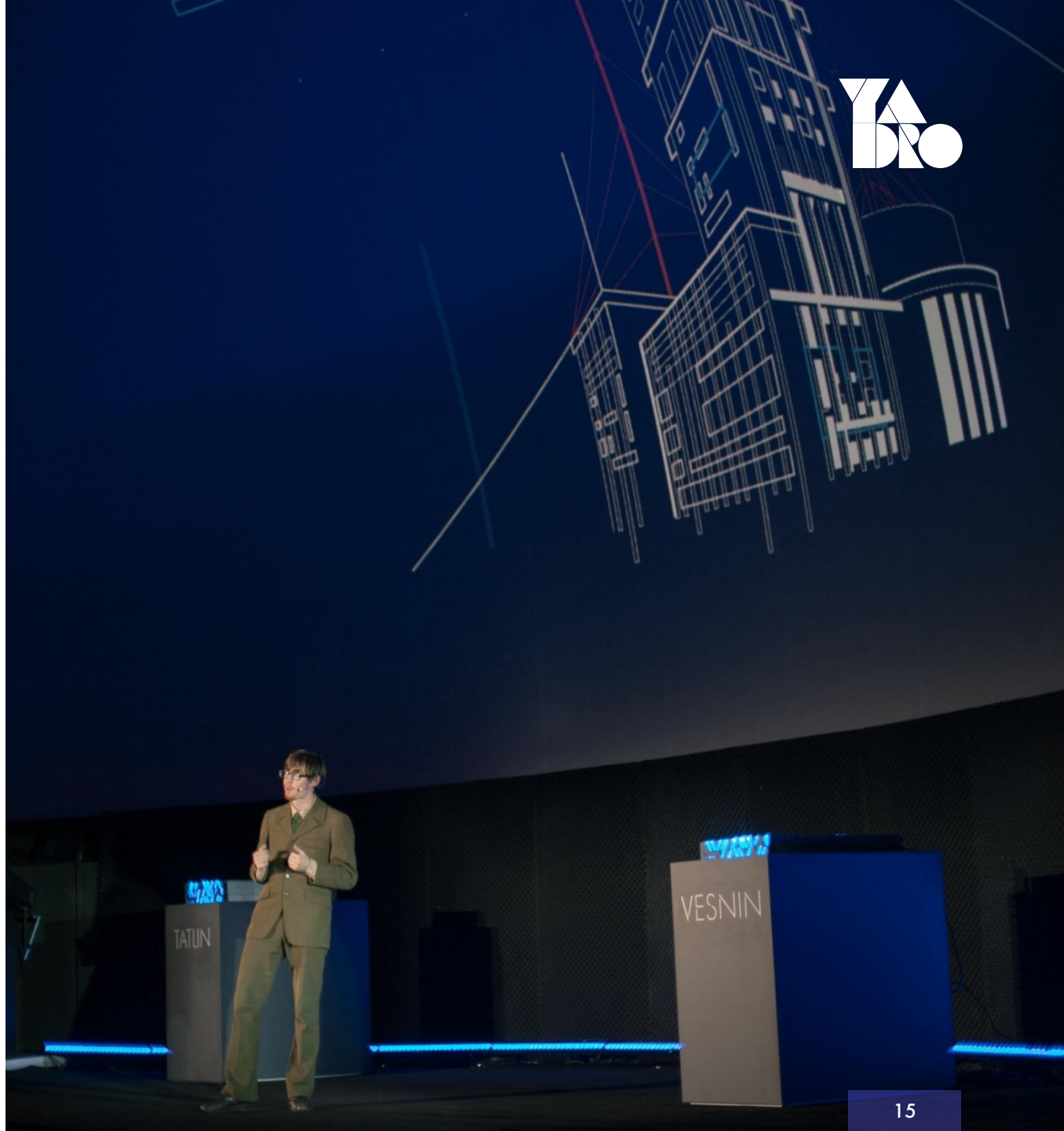
- Стимул — качество и надежность продуктов
- «Гильдия» как кросс-проектная структура
- Слона едим по кусочкам...
- «Не кошмарьте разработчиков!»

Человеческий фактор

- Архитектор безопасности — что за зверь и как его добыть
- Кто такой Security Champion?
- А кто еще нужен?
- «Ахтунг! Олды в чате!»
- Интерес, безразличие и мотивация
- «Сферический разработчик в вакууме» и его мутации — «Примадонна», «Хранитель сакрального знания»

Заключение

- SDL это хорошо, но без фундамента — никак
- Качество и надежность — на первом месте
- Инженеров надо выращивать



yadro.com

123376, Москва г., Рочдельская ул., дом 15, строение 15

a.dubinin@yadro.com