Документация как код? Это просто!

Александр Дубинин
Группа компаний YADRO

О компании ЯДРО

YADRO сейчас — российская технологическая компания, 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.

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

Продукты ЯДРО: yadro.com/ru/products

Что новенького?

Составляющие качественного продукта это:

- Инженерная культура и образование (продолжаем, это фундамент!)

Что новенького?

Составляющие качественного продукта это:

- Инженерная культура и образование (продолжаем, это фундамент!)
- Проработанная архитектура (***было в прошлый раз*** :)

Что новенького?

Составляющие качественного продукта это:

- Инженерная культура и образование (продолжаем, это фундамент!)
- Проработанная архитектура (*было в прошлый раз* :)
- Знаем каждый байт (SCA)
- Понятный, правильный и красивый код (SAST)
- Тестирование (DAST, Q&A)

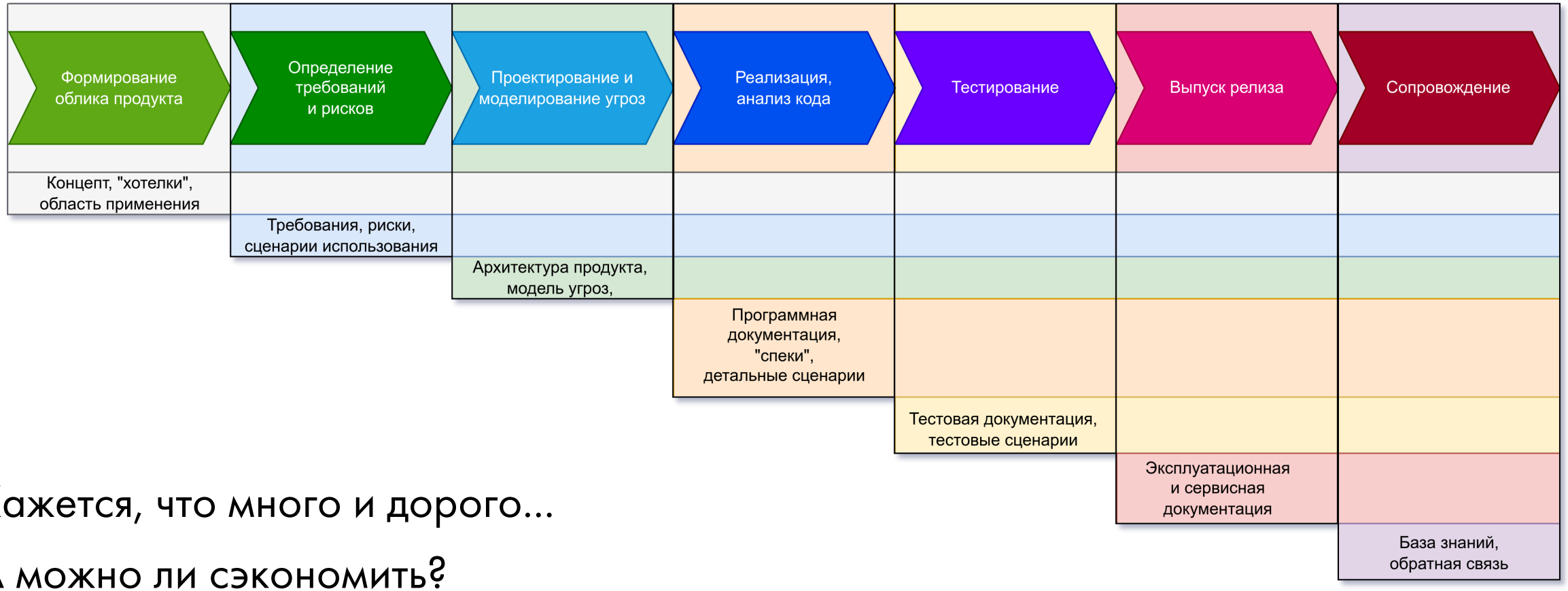
Что новенького?

Составляющие качественного продукта это:

- Инженерная культура и образование (продолжаем, это фундамент!)
- Проработанная архитектура (*было в прошлый раз* :)
- Знаем каждый байт (SCA)
- Понятный, правильный и красивый код (SAST)
- Тестирование (DAST, Q&A)
- **И документация - цемент, который скрепляет всё вместе!**

Документация как существенная часть продукта

Документация в жизненном цикле продукта с SDL:



Кажется, что много и дорого...

А можно ли сэкономить?

Как делают обычно? ФФП!

Особенности *обычного* процесса производства коммерческих программных продуктов (процесс ФФП: «фигак-фигак-продакшен»):

- Функциональность на 1 месте, релизы «выпекаются» как пирожки;
- Тестирование: «Ну это дорого, может лучше на клиентах потестим?»;
- Программная документация - «а давайте потом, как нибудь в другой раз...»

Как делают обычно? ФФП!

Особенности *обычного* процесса производства коммерческих программных продуктов (процесс ФФП: «фигак-фигак-продакшен»):

- Функциональность на 1 месте, релизы «выпекаются» как пирожки;
- Тестирование: «Ну это дорого, может лучше на клиентах потестим?»;
- Программная документация - «а давайте потом, как нибудь в другой раз...»

Результат - сиюминутная прибыль/экономия, но риски и расходы потом:

- Лавинообразный рост конфликтующего функционала;
- Спонтанные проблемы с безопасностью и стабильностью продукта;
- «Сферические разработчики в ваккууме» и «носители сакрального знания»;
- Длительное время обучения новых инженеров.

А что если с SDL?

Процесс SSDLC - часть инженерной культуры, для продукта:

- Просто и понятно **документированы** «хотелки» (в т.ч. ИБ), «облик продукта»;

А что если с SDL?

Процесс SSDLC - часть инженерной культуры, для продукта:

- Просто и понятно **документированы** «хотелки» (в т.ч. ИБ), «облик продукта»;
- Оценены риски, угрозы, **расписаны** сценарии использования (use cases);

А что если с SDL?

Процесс SSDLC - часть инженерной культуры, для продукта:

- Просто и понятно **документированы** «хотелки» (в т.ч. ИБ), «облик продукта»;
- Оценены риски, угрозы, **расписаны** сценарии использования (use cases);
- Архитектура (HLA) спроектирована осознанно и **документирована**, не имеет функциональной избыточности;

А что если с SDL?

Процесс SSDLC - часть инженерной культуры, для продукта:

- Просто и понятно **документированы** «хотелки» (в т.ч. ИБ), «облик продукта»;
- Оценены риски, угрозы, **расписаны** сценарии использования (use cases);
- Архитектура (HLA) спроектирована осознанно и **документирована**, не имеет функциональной избыточности;
- Проработан технический проект (HLD), каждая функциональность имеет **спецификацию** и связана со сценариями использования;

А что если с SDL?

Процесс SSDLC - часть инженерной культуры, для продукта:

- Просто и понятно **документированы** «хотелки» (в т.ч. ИБ), «облик продукта»;
- Оценены риски, угрозы, **расписаны** сценарии использования (use cases);
- Архитектура (HLA) спроектирована осознанно и **документирована**, не имеет функциональной избыточности;
- Проработан технический проект (HLD), каждая функциональность имеет **спецификацию** и связана со сценариями использования;
- Код ясен и понятен (**документирован**), не имеет дефектов (SAST);

А что если с SDL?

Процесс SSDLC - часть инженерной культуры, для продукта:

- Просто и понятно **документированы** «хотелки» (в т.ч. ИБ), «облик продукта»;
- Оценены риски, угрозы, **расписаны** сценарии использования (use cases);
- Архитектура (HLA) спроектирована осознанно и **документирована**, не имеет функциональной избыточности;
- Проработан технический проект (HLD), каждая функциональность имеет **спецификацию** и связана со сценариями использования;
- Код ясен и понятен (**документирован**), не имеет дефектов (SAST);
- Тестировщики понимают, что и как они тестируют (DAST) (**use cases!**);

А что если с SDL?

Процесс SSDLC - часть инженерной культуры, для продукта:

- Просто и понятно **документированы** «хотелки» (в т.ч. ИБ), «облик продукта»;
- Оценены риски, угрозы, **расписаны** сценарии использования (use cases);
- Архитектура (HLA) спроектирована осознанно и **документирована**, не имеет функциональной избыточности;
- Проработан технический проект (HLD), каждая функциональность имеет **спецификацию** и связана со сценариями использования;
- Код ясен и понятен (**документирован**), не имеет дефектов (SAST);
- Тестировщики понимают, что и как они тестируют (DAST) (**use cases!**);
- Инженеры не бегают за «носителями сакрального знания», а **читают доку**, учатся и **продуктивно делают свою работу** :)

Задачка по управлению проектом:

Дано:

- Инженеры очень не любят писать документацию.

Задачка по управлению проектом:

Дано:

- Инженеры очень не любят писать документацию.
- Инженеры умеют (обычно ;) писать код.

Задачка по управлению проектом:

Дано:

- Инженеры очень не любят писать документацию.
- Инженеры умеют (обычно ;) писать код.
- Инженеры (и архитекторы тоже :) ненавидят офисные пакеты!

Задачка по управлению проектом:

Дано:

- Инженеры очень не любят писать документацию.
- Инженеры умеют (обычно ;) писать код.
- Инженеры (и архитекторы тоже :) ненавидят офисные пакеты!
- Техпис не сделает контент за инженера.

Задачка по управлению проектом:

Дано:

- Инженеры очень не любят писать документацию.
- Инженеры умеют (обычно ;) писать код.
- Инженеры (и архитекторы тоже :) ненавидят офисные пакеты!
- Техпис не сделает контент за инженера.

Решение:

Инженер пишет документацию, но как код!

Документация как код: особенности

- Разделение оформления и содержания.

Документация как код: особенности

- Разделение оформления и содержания;
- Контент создается сразу, по мере работы.

Документация как код: особенности

- Разделение оформления и содержания;
- Контент создается сразу, по мере работы;
- Результат выдается в любом формате.

Документация как код: особенности

- Разделение оформления и содержания;
- Контент создается сразу, по мере работы;
- Результат выдается в любом формате;
- ГОСТ ЕСПД - всего лишь один из выходных форматов.

Документация как код: особенности

- Разделение оформления и содержания;
- Контент создается сразу, по мере работы;
- Результат выдается в любом формате;
- ГОСТ ЕСПД - всего лишь один из выходных форматов;
- Процесс генерации документации автоматизирован.

Документация как код: особенности

- Разделение оформления и содержания;
- Контент создается сразу, по мере работы;
- Результат выдается в любом формате;
- ГОСТ ЕСПД - всего лишь один из выходных форматов;
- Процесс генерации документации автоматизирован;
- Один DocOps инженер вместо полка «техников-оформителей» и недовольных разработчиков.

Что такое DocOps?

- Как и DevOps - **процесс** работы с документацией в CI/CD;
- Инженеры, автоматизирующие процесс, понимающие нужные преобразования форматов.

Как это все работает вместе?

- Контент создается в Markdown;

Как это все работает вместе?

- Контент создается в Markdown;
- Диаграммы создаются как код (diagrams, plantuml, drawio etc.);

Как это все работает вместе?

- Контент создается в Markdown;
- Диаграммы создаются как код (diagrams, plantuml, drawio, etc.);
- Всё хранится в системе контроля версий;

Как это все работает вместе?

- Контент создается в Markdown;
- Диаграммы создаются как код (diagrams, plantuml, drawio, etc.);
- Всё хранится в системе контроля версий;
- Инженер пишет код - и если его поведение изменилось, сразу меняет документацию (например, в том-же GIT репозитории, каталог «docs»);

Как это все работает вместе?

- Контент создается в Markdown;
- Диаграммы создаются как код (diagrams, plantuml, drawio, etc.);
- Всё хранится в системе контроля версий;
- Инженер пишет код - и если его поведение изменилось, сразу меняет документацию (например, в том-же GIT репозитории, каталог «docs»);
- Каждая сборка формирует документацию, актуальную для собираемой версии.

Инструменты: Markdown

- Просто текст - для инженеров!
- Невероятно популярен
- Элементарно расширяется
- Open Source

```
27  # Тест стилей в MD
28
29  ## Заголовок второго уровня
30
31  Скажем миру "Здрасте!":
32
33      #!/bin/bash
34      printf "%s\n" "Hello, world!"
35
36  ### Заголовок третьего уровня
37
38  1. Элемент списка 1
39  2. Элемент списка 2
40  3. Элемент списка 3
41      1. Элемент списка 3.1
42          1. Элемент списка 3.1.1
43      2. Элемент списка 3.2
```

Тест стилей в MD

1. Заголовок второго уровня

Скажем миру "Здрасте!":

```
#!/bin/bash
printf "%s\n" "Hello, world!"
```

1.1. Заголовок третьего уровня

1. Элемент списка 1
2. Элемент списка 2
3. Элемент списка 3
    3.1. Элемент списка 3.1
        3.1.1. Элемент списка 3.1.1
    3.2. Элемент списка 3.2

Инструменты: DocBook

- XML (обычный текст!)
- Есть готовые стили ГОСТ ЕСПД (lab50.ru, огромное спасибо!)
- Open Source

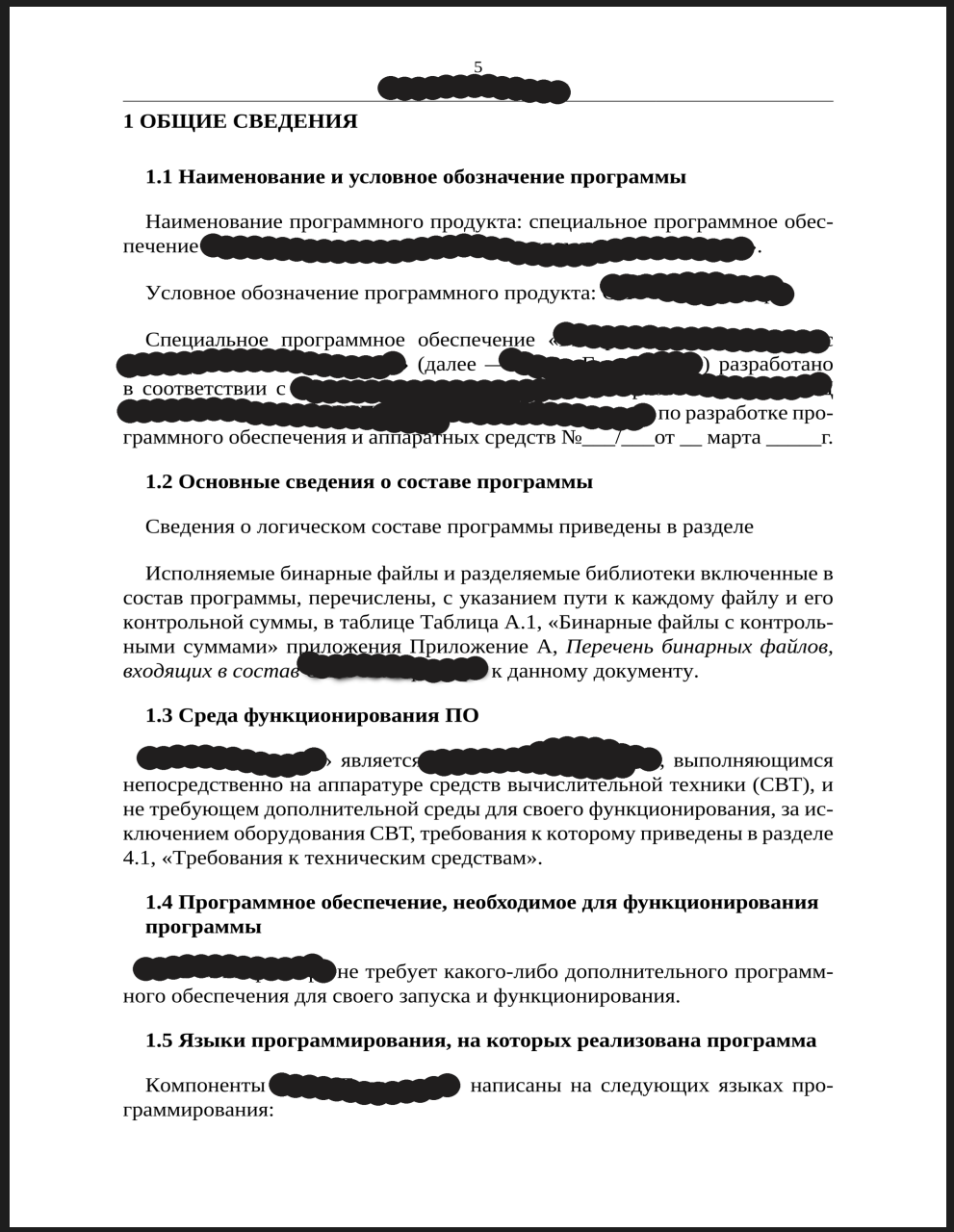
```
<title>Общие сведения</title>

<!-- ГОСТ: обозначение и наименование программы -->
<xi:include href="shared/Oboznachenie.xml"/>

<!-- РД НДВ, 3.2.1:  основные сведения о составе (с указанием контрольных сумм файлов,
  входящих в состав ПО) -->
<section>
  <title>Основные сведения о составе программы</title>
  <para>Сведения о логическом составе программы приведены в разделе </para>
  <para>Исполняемые бинарные файлы и разделяемые библиотеки включенные в состав программы,
    перечислены, с указанием пути к каждому файлу и его контрольной суммы, в таблице <xref
      linkend="bin_files_checksums"/> приложения <xref linkend="Binary.Files"/> к данному
    документу.</para>
</section>

<section>
  <title>Среда функционирования ПО</title>
  <para>&product; [redacted], выполняющимся непосредственно на аппаратуре
    средств вычислительной техники (СВТ), и не требующем дополнительной среды для своего
    функционирования, за исключением оборудования СВТ, требования к которому приведены в разделе
      <xref linkend="shared.Requirements.Tech" xrefstyle="select: labelnumber quotedtitle"
    />.</para>
</section>

<section>
  <!-- программное обеспечение, необходимое для функционирования программы -->
  <title>Программное обеспечение, необходимое для функционирования программы</title>
```



5 [redacted]

1 ОБЩИЕ СВЕДЕНИЯ

1.1 Наименование и условное обозначение программы

Наименование программного продукта: специальное программное обеспечение [redacted].

Условное обозначение программного продукта: [redacted]

Специальное программное обеспечение «[redacted]» (далее — [redacted]) разработано в соответствии с [redacted] по разработке программного обеспечения и аппаратных средств №___/___от __ марта _____г.

1.2 Основные сведения о составе программы

Сведения о логическом составе программы приведены в разделе

Исполняемые бинарные файлы и разделяемые библиотеки включенные в состав программы, перечислены, с указанием пути к каждому файлу и его контрольной суммы, в таблице Таблица А.1, «Бинарные файлы с контрольными суммами» приложения Приложение А, *Перечень бинарных файлов, входящих в состав* [redacted] к данному документу.

1.3 Среда функционирования ПО

[redacted] является [redacted], выполняющимся непосредственно на аппаратуре средств вычислительной техники (СВТ), и не требующем дополнительной среды для своего функционирования, за исключением оборудования СВТ, требования к которому приведены в разделе 4.1, «Требования к техническим средствам».

1.4 Программное обеспечение, необходимое для функционирования программы

[redacted] не требует какого-либо дополнительного программного обеспечения для своего запуска и функционирования.

1.5 Языки программирования, на которых реализована программа

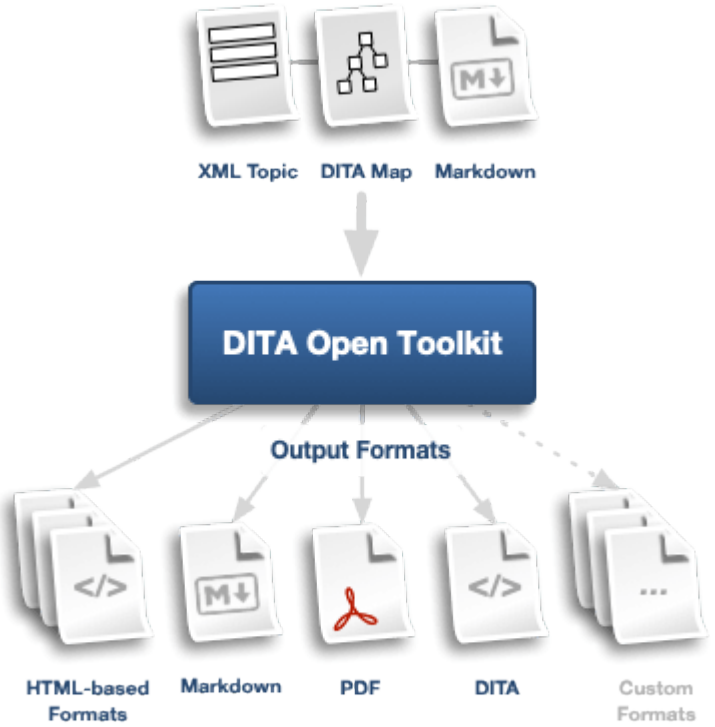
Компоненты [redacted] написаны на следующих языках программирования:

Инструменты: dita-ot (DITA)

«Всеядный» процессор DITA. Можно использовать:

- DITA XML («родной» формат)
- LWDITA («облегченный» XML)
- Markdown
- reStructured (основной формат Sphynx)
- HTML
- Все что «переварят» плагины к dita-ot
- Скрипты на bash и python :)

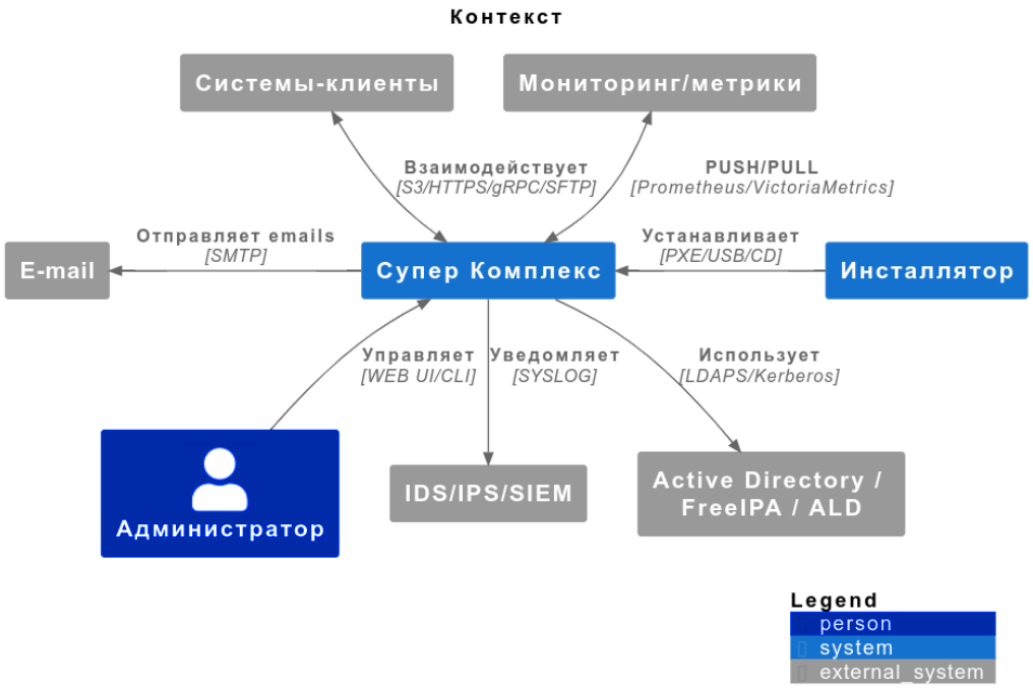
Множество учебников и ресурсов в Сети.



https://www.dita-ot.org/

Диаграммы как код

- Диаграммы как код
- Автоматическая генерация SVG
- Контроль версий (GIT)
- C4, UML, BPML - что угодно!




```
@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
```

Другие источники контента

Контент для документации также можно получить:

- Из аннотаций в коде и комментариев:
  - Doxygen (C, C++, C#, Java, Perl, PHP, Python, и т.д.)
  - Javadoc (промышленный стандарт для Java)
  - JSDoc (для JavaScript)
  - Sphinx (стандарт для Python)
- Из CSV файлов (большие таблицы);
- Из всего, что можно обрабатывать как текст.

Множество статей и других ресурсов в Сети.

```
#!/bin/bash
# Сканируем все файлы:
BASEDIR=`pwd`
cat packages.txt | sed 's@:amd64@@g' |
while read line ; do
    name=`echo $line | awk '{print $2}'`
    version=`echo $line | awk '{print $3}'`
    printf "<row><entry>$name</entry><entry>$version</entry></row>\n"
done >content_be.xml
# Заголовок DocBook
cat $BASEDIR/header_be.xml > $BASEDIR/BuildEnv.Packages.xml
# Список файлов:
cat $BASEDIR/content_be.xml >> $BASEDIR/BuildEnv.Packages.xml
# Окончание DocBook
cat $BASEDIR/footer.xml >> $BASEDIR//BuildEnv.Packages.xml
exit 0
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE section [
<!ENTITY % gentities SYSTEM "../project.entities">
%gentities;
]>
<section version="5.0" xml:id="Buildenv.Packages" xml:lang="ru"
  xmlns="http://docbook.org/ns/docbook" xmlns:xlink="http://www.w3.org/1999/x
  xmlns:xi="http://www.w3.org/2001/XInclude" xmlns:svg="http://www.w3.org/200
  xmlns:m="http://www.w3.org/1998/Math/MathML" xmlns:html="http://www.w3.org/
  xmlns:db="http://docbook.org/ns/docbook">
  <title>Перечень пакетов программной среды технологического оснащения</title>

  <para>В таблице ниже перечислены версии пакетов, установленных в среде сборк
  построении &product;.</para>
  <para>Основой программного обеспечения средств технологического оснащения яв

  <table xml:id="buildenv_packages" tabstyle="small">
    <title>Пакеты среды сборки средств технологического оснащения</title>
    <tgroup cols="2" align="left">
      <colspec colname="col1" colwidth="1*"/>
      <colspec colname="col2" colwidth="1*"/>
      <thead>
        <row>
          <entry align="center">Название пакета</entry>
          <entry align="center">Версия пакета</entry>
        </row>
```

Генерируем результат

Из разных форматов и источников формируем документы:

- С помощью простого GNU Make;
- С помощью mkdocs, pandoc, dita-ot и еще множества вариантов.

```
13 %.xml : $(COMMON_DEPS)
14
15 %.fop : %.xml
16   java -cp "/usr/share/java/saxon.jar:/usr/share/java/xslthl-2.1.3.jar:/usr/share/java/xml-commons-resolver-1.1.jar
17     -Djavax.xml.parsers.DocumentBuilderFactory=org.apache.xerces.jaxp.DocumentBuilderFactoryImpl \
18     -Djavax.xml.parsers.SAXParserFactory=org.apache.xerces.jaxp.SAXParserFactoryImpl \
19     -Dorg.apache.xerces.xni.parser.XMLParserConfiguration=org.apache.xerces.parsers.XIncludeParserConfiguration \
20     -Dxslthl.config="file:///usr/share/xml/docbook/stylesheet/docbook-xsl/highlighting/xslthl-config.xml" \
21     com.icl.saxon.StyleSheet \
22     -x org.apache.xml.resolver.tools.ResolvingXMLReader \
23     -y org.apache.xml.resolver.tools.ResolvingXMLReader \
24     -r org.apache.xml.resolver.tools.CatalogResolver \
25     -o $@ $< $(XSL_STYLESHEET) $(DOCUMENT_PROFILE)\
26     use.extensions=1 espd.decimal='$(BASE_DECIMAL)-$(RELEASE) $(DOCUMENT)'
27
28 %.pdf : %.fop
29   fop -c $(FOP_CONFIG) -fo $<  -pdf $@
```

Заключение

- Процесс создания документации - неотъемлемая часть процесса РБПО;
- Если документацию создавать как код и по мере разработки - это не затратно и не «больно» для инженеров;
- Документируем все этапы создания продукта;
- Хорошая документация:
  - Повышает продуктивность работы команды;
  - Служит индикатором качества продукта.

yadro.com

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

a.dubinin@yadro.com