Лекции 11-12
«**Проектирование конфигурационного управления**»

Овчинников П.Е.
МГТУ «СТАНКИН»,
ст.преподаватель кафедры ИС

13.03.2019

Дальше

Лекция 11
«**Управление требованиями, изменениями, инцидентами**»

Овчинников П.Е.
МГТУ «СТАНКИН»,
ст.преподаватель кафедры ИС

06.02.2019

Новые образовательные программы

ПРОФЕССИОНАЛЬНЫЙ СТАНДАРТ

Руководитель проектов в области информационных технологий

154

Регистрационный номер

I. Общие сведения о профессии

Менеджмент проектов в области информационных технологий (ИТ) | 06.016

(наименование вида профессиональной деятельности) | Код

Основная цель вида профессиональной деятельности:

Менеджмент проектов в области ИТ (планирование, организация исполнения, контроль и анализ отклонений для эффективного достижения целей проекта в рамках утвержденных заказчиком требований, бюджета и сроков

Новые образовательные программы

Обобщенные трудовые функции			Трудовые функции
код	наименование	уровень квалификации	наименование
A	Управление проектами в области ИТ на основе полученных планов проектов в условиях, когда проект не выходит за пределы утвержденных параметров	6	Идентификация конфигурации информационной системы (ИС) в соответствии с полученным планом
			Ведение отчетности по статусу конфигурации ИС в соответствии с полученным планом
			Аудит конфигураций ИС в соответствии с полученным планом
			Организация репозитория проекта в области ИТ в соответствии с полученным планом
			Проверка реализации запросов на изменение (верификация) в соответствии с полученным планом
			Организация заключения договоров в проектах в соответствии с трудовым заданием
			Мониторинг выполнения договоров в проектах в области ИТ в соответствии с полученным планом
			Организация заключения дополнительных соглашений к договорам в соответствии с трудовым заданием

Новые образовательные программы

Требования к образованию и обучению	Высшее образование – программы бакалавриата Повышение квалификации в области проектного менеджмента
Требования к опыту практической работы	Рекомендуется: работа по профессиям «системный аналитик» и/или «архитектор программного обеспечения» не менее одного года
Особые условия допуска к работе	-

Дополнительные характеристики

Наименование документа	Код	Наименование базовой группы, должности (профессии) или специальности
ОКЗ	1236	Руководители подразделений (служб) компьютерного обеспечения
ЕКС[3]	-	Инженер-программист (программист)
ОКСО[4]	010400	Информационные технологии
	010500	Прикладная математика и информатика
	080700	Бизнес-информатика
	080800	Прикладная информатика
	230100	Информатика и вычислительная техника
	230200	Информационные системы

Новые образовательные программы

3.1.1. Трудовая функция

Наименование	Идентификация конфигурации ИС в соответствии с полученным планом	Код	A/01.6	Уровень (подуровень) квалификации	6

Происхождение трудовой функции	Оригинал X	Заимствовано из оригинала	Код оригинала	Регистрационный номер профессионального стандарта

Трудовые действия	Определение базовых элементов конфигурации ИС
	Присвоение версии базовым элементам конфигурации ИС
	Установление базовых версий конфигурации ИС
Необходимые умения	Работать с системой контроля версий
	Анализировать входные данные
Необходимые знания	Основы конфигурационного управления
Другие характеристики	-

Терминология: конфигурация

ГОСТ Р ИСО 10007-2007 Менеджмент организации.
РУКОВОДЯЩИЕ УКАЗАНИЯ ПО УПРАВЛЕНИЮ КОНФИГУРАЦИЕЙ

конфигурация (configuration):
взаимосвязанные функциональные и физические **характеристики** продукции, **установленные в данных** о конфигурации продукции

элемент конфигурации (configuration item):
объект конфигурации, выполняющий **законченную функцию**

управление конфигурацией (configuration management):
скоординированные действия, направленные на **формирование** и **контроль** конфигурации

Примечание - Управление конфигурацией обычно включает в себя поддержку технической и административной деятельности, связанной с управлением продукцией и требованиями к ее конфигурации на всех стадиях жизненного цикла продукции.

ГОСТ Р ИСО 10007-2007

Терминология: конфигурация

ГОСТ Р ИСО 10007-2007 Менеджмент организации.
РУКОВОДЯЩИЕ УКАЗАНИЯ ПО УПРАВЛЕНИЮ КОНФИГУРАЦИЕЙ

Данные о конфигурации продукции

Данные о конфигурации продукции включают в себя **описание продукции** и ее **эксплуатационные характеристики**

Обычно данные о конфигурации продукции включают в себя:

- **требования**
- технические условия
- проектную документацию
- **перечень составных частей**
- **документацию** на программное обеспечение
- **модели**
- требования к испытаниям
- руководство по техническому обслуживанию и эксплуатации

ГОСТ Р ИСО 10007-2007

Терминология: требования

ГОСТ Р ИСО/ТС 10303-1348-2014 Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными. Часть 1348. Прикладной модуль. Управление требованиями

Требования настоящего стандарта распространяются на:

- представление данных, идентифицирующих требование и его версии
- задание требования для изделия или другого объекта
- идентификацию источника требования
- взаимосвязи между требованиями
- определение версии требования, подходящей для одной или нескольких прикладных областей и одной или нескольких стадий жизненного цикла
- идентификацию предметной области, соответствующей описанию требований
- взаимосвязи между требованиями, выраженными на уровне определения представлений

ГОСТ Р ИСО/ТС 10303-1348-2014

Терминология: требования

ГОСТ Р ИСО/ТС 10303-1348-2014 Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными. Часть 1348. Прикладной модуль. Управление требованиями

- задание информации о дате и времени требованиям, определениям требований, версиям требований, взаимосвязям между версиями требований, источникам требований, взаимосвязям между группами требований, взаимосвязям между отслеживаниями выполнения требований и заданиям требований
- задание информации об утверждении требованиям, определениям требований, версиям требований, взаимосвязям между версиями требований, источникам требований, взаимосвязям между группами требований, взаимосвязям между отслеживаниями выполнения требований и заданиям требований
- задание информации об идентификации и идентификации альтернативных имен требованиям, определениям требований, версиям требований, взаимосвязям между версиями требований, источникам требований, взаимосвязям между группами требований, взаимосвязям между отслеживаниями выполнения требований и заданиям требований

ГОСТ Р ИСО/ТС 10303-1348-2014

Терминология: требования

ГОСТ Р ИСО/ТС 10303-1348-2014 Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными. Часть 1348. Прикладной модуль. Управление требованиями

- задание информации о сотруднике и организации требованиям, определениям требований, версиям требований, взаимосвязям между версиями требований, источникам требований, взаимосвязям между группами требований, взаимосвязям между отслеживаниями выполнения требований и заданиям требований
- задание информации о контракте требованиям, определениям требований, версиям требований, взаимосвязям между версиями требований, источникам требований, взаимосвязям между группами требований, взаимосвязям между отслеживаниями выполнения требований и заданиям требований
- задание информации об уровне секретности требованиям, определениям требований, версиям требований, взаимосвязям между версиями требований, источникам требований, взаимосвязям между группами требований, взаимосвязям между отслеживаниями выполнения требований и заданиям требований

ГОСТ Р ИСО/ТС 10303-1348-2014

Терминология: требования

ГОСТ Р ИСО/ТС 10303-1348-2014 Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными. Часть 1348. Прикладной модуль. Управление требованиями

- задание информации о языке требованиям, определениям требований, версиям требований, взаимосвязям между версиями требований, источникам требований, взаимосвязям между группами требований, взаимосвязям между отслеживаниями выполнения требований и заданиям требований

USE FROM Classification_assignment_arm; - - ISO/TS 10303-1114

USE FROM Collection_identification_and_version_arm; - - ISO/TS 10303-1396

USE FROM Document_assignment_arm; - - ISO/TS 10303-1122

USE FROM Document_properties_arm; - - ISO/TS 10303-1126

USE FROM Effectivity_application_arm; - - ISO/TS 10303-1059

USE FROM Identification_relationship_arm; - - ISO/TS 10303-1398

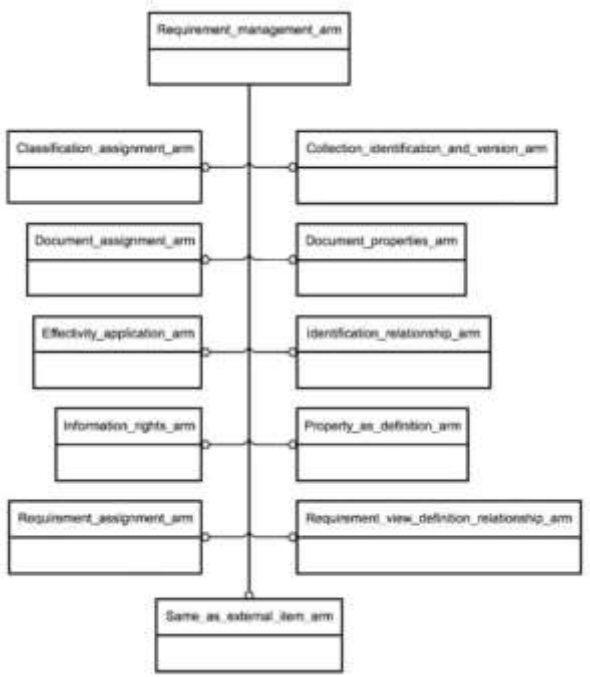
USE FROM Information_rights_arm; - - ISO/TS 10303-1241

USE FROM Property_as_definition_arm; - - ISO/TS 10303-1399

USE FROM Requirement_assignment_arm; - - ISO/TS 10303-1233

USE FROM Requirement_view_definition_relationship_arm; - - ISO/TS 10303-1142

USE FROM Same_as_external_item_arm; - - ISO/TS 10303-1402



ГОСТ Р ИСО/ТС 10303-1348-2014

Терминология: изменения

ГОСТ Р ИСО 10007-2007 Менеджмент организации.
РУКОВОДЯЩИЕ УКАЗАНИЯ ПО УПРАВЛЕНИЮ КОНФИГУРАЦИЕЙ

5.4 Управление изменениями

После **первоначального** установления данных о конфигурации продукции необходимо **управлять всеми изменениями** конфигурации продукции

Потенциальное воздействие изменений, требований потребителя и базовой конфигурации влияют на степень управления, необходимую для введения предложенного изменения или применения разрешения на отклонение

Процесс управления изменением должен быть документально оформлен и должен включать в себя:

- **описание** процесса, оправдательные документы и записи об изменении
- **классификацию** изменения с точки зрения его сложности, необходимых ресурсов и планирования для выполнения
- оценку **последствий** изменения
- подробное описание того, как изменение должно быть **подготовлено**
- подробное описание того, как изменение должно быть **выполнено** и **верифицировано**

ГОСТ Р ИСО 10007-2007

Терминология: инциденты

ITIL (произносится как *«айти́л»*, англ. *IT Infrastructure Library* — библиотека инфраструктуры информационных технологий) — **библиотека**, описывающая **лучшие** из применяемых на **практике** способов организации работы подразделений или компаний, занимающихся предоставлением услуг в области информационных технологий

Управления инцидентами по ITIL – это **процесс**, направленный на **устранение** каких-либо инцидентов, вызывающих прерывания ИТ-услуг, самым быстрым и эффективным способом

Согласно ITIL **инцидентом** является как **неисправность** в работе **программного** или **аппаратного** обеспечения, так и любое **отклонение** от согласованных с пользователем **параметров** предоставления ИТ-услуги, которое приводит к ее **прерыванию** или **снижению качества**

Процесс управления инцидентами акцентируется исключительно на устранении **последствий** сбоев в ИТ-услугах. Поиском и анализом **причин** возникновения инцидентов занимается процесс **управления проблемами**

Терминология: инциденты

Основные задачи процесса управления инцидентами:

- Выявление и регистрация сбоев в предоставлении ИТ-услуг
- Классификация инцидентов
- Назначение задач ИТ-персоналу, отвечающему за восстановление этих ИТ-услуг
- Контроль соответствия времени закрытия инцидентов параметрам, определенным в соглашении об уровне сервиса (SLA)

- ID
- Категория
- Срочность
- Влияние
- Время регистрации
- Регистратор
- Метод информирования об инциденте
- Данные о пользователе
- Метод обратной связи

- Описание симптомов
- Статус
- Связанные КЕ
- Группа поддержки
- Связанная проблема/известная ошибка
- Предпринятые действия
- Время решения
- Код закрытия
- Время закрытия

ITIL

Терминология: реагирование

- **Обнаружение.** Пользователи услуг или системы мониторинга обнаруживают и сообщают об ошибке, сбое или прерывании ИТ-сервиса.
- **Регистрация.** Инцидент регистрируется, вносится детализация: дата и время события, информация об источнике сообщения, уникальный ID и др.
- **Категоризация.** Определяется категория инцидента с учетом преднастроенных значений.
- **Приоритизация.** Назначается приоритет, степень воздействия, срочность.
- **Диагностика.** Проводится первичное расследование, фиксируются признаки сбоя и результаты диагностики причин, устанавливается «диагноз».
- **Эскалация.** В сложных случаях или с учетом заранее настроенных условий инцидент передается специалистам второй линии поддержки, в т.ч. оповещается менеджер, бизнес-заказчик и т.п.
- **Решение и восстановление.** Прерывание обслуживания устраняется, проверяется успешность выполнения. Проводится документирование, чтобы в будущем при возникновении подобных событий использовать накопленные знания для более быстрой диагностики и решения.
- **Закрытие.** Обращение закрывается. Пользователь подтверждает, что уровень сервиса восстановлен. В противном случае, работы по инциденту возобновляются.

ITIL

Информационная безопасность

ГОСТ Р ИСО/МЭК 27000-2012 Информационная технология. Методы и средства обеспечения безопасности. Системы менеджмента информационной безопасности. Общий обзор и терминология

Информационная безопасность включает в себя три основных измерения:
- **конфиденциальность**,
- **доступность** и
- **целостность**.

С целью обеспечения длительного непрерывного успеха в бизнесе и уменьшения нежелательных воздействий информационная безопасность предусматривает применение соответствующих мер безопасности, которые включают в себя рассмотрение широкого диапазона угроз, а также управление этими мерами.

Информационная безопасность достигается посредством применения соответствующего набора средств управления, определенного с помощью **процесса управления рисками** и управляемого с использованием СМИБ, включая политику, процессы, процедуры, организационные структуры, программное и аппаратное обеспечение, чтобы защитить идентифицированные информационные активы.

Стандарты в области ИБ *http://docs.cntd.ru/document/1200102762*

Терминология: кибербезопасность

ГОСТ Р МЭК 62443-2-1-2015 Сети коммуникационные промышленные. Защищенность (кибербезопасность) сети и системы. Часть 2-1. Составление программы обеспечения защищенности (кибербезопасности) системы управления и промышленной автоматики

Организации, применяющие IACS (системы промышленной автоматики и контроля), начали применять готовые коммерческие технологии (COTS), разработанные для бизнес-систем, используемых в их повседневных процессах, в результате чего возрос риск кибератак, направленных на оборудование IACS. Как правило, такие системы в среде IACS по многим причинам не настолько робастны, как системы, специально спроектированные как IACS для подавления кибератак. Подобные недостатки могут привести к последствиям, которые отразятся на уровне охраны труда, промышленной безопасности и охраны окружающей среды (HSE).

система управления кибербезопасностью (cyber security management system): Программа, разработанная организацией для поддержания кибербезопасности всех имущественных объектов данной организации на заданном уровне конфиденциальности, целостности и доступности, независимо от того, относятся ли данные объекты к бизнес-процессам или системам IACS организации.

http://docs.cntd.ru/document/1200102762

Терминология: аутентификация

ГОСТ Р ИСО/МЭК 27000-2012 Информационная технология. Методы и средства обеспечения безопасности. Системы менеджмента информационной безопасности. Общий обзор и терминология

2.5 **аутентификация (authentication):** Обеспечение гарантии того, что заявленные характеристики объекта правильны.

Р 50.1.056-2005 Техническая защита информации. Основные термины и определения

3.5.11 **аутентификация (подлинности субъекта доступа):** Действия по проверке подлинности субъекта доступа в информационной системе

ГОСТ Р 52633.0-2006 Защита информации. Техника защиты информации. Требования к средствам высоконадежной биометрической аутентификации

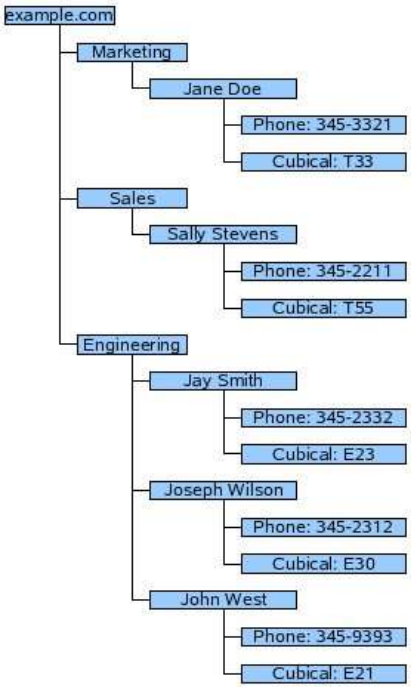
3.6 **биометрическая идентификация:** Преобразование совокупности примеров биометрических образов человека, позволяющее описать их стационарную и случайную составляющие, например, в виде математического ожидания и дисперсий контролируемых параметров или, например, в виде параметров обученной сети искусственных нейронов

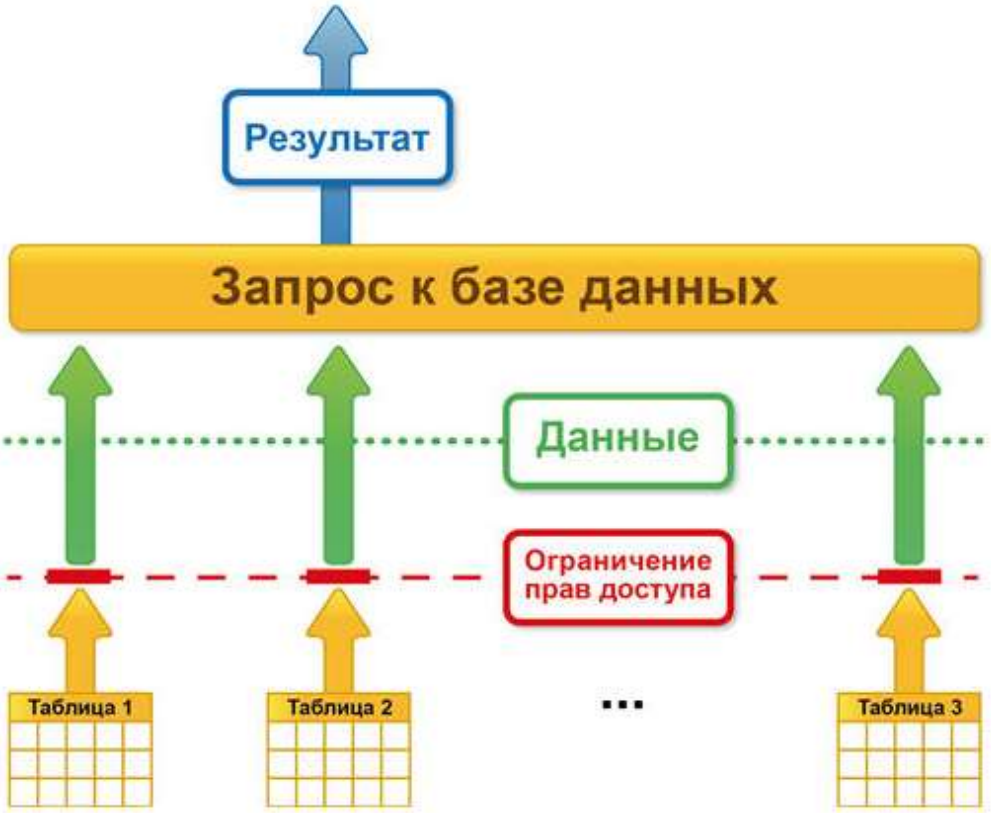
http://docs.cntd.ru/document/1200102762
http://docs.cntd.ru/document/1200048922
Биометрическая аутентификация

Терминология: авторизация

Р 50.1.056-2005 Техническая защита информации. Основные термины и определения

3.5.10 **санкционирование доступа; *авторизация*:** Предоставление субъекту прав на доступ, а также предоставление доступа в соответствии с установленными правами на доступ





http://docs.cntd.ru/document/1200044768
http://v8.1c.ru/overview/Term_000000579.htm
https://ru.wikipedia.org/wiki/Active_Directory

Терминология: угрозы

ГОСТ Р ИСО/МЭК 27000-2012 Информационная технология. Методы и средства обеспечения безопасности. Системы менеджмента информационной безопасности. Общий обзор и терминология

2.45 **угроза** (threat): Возможная причина нежелательного инцидента, который может нанести ущерб системе или организации.

2.4 **атака** (attack): Попытка уничтожения, раскрытия, изменения, блокирования, кражи, получения несанкционированного доступа к **активу** (2.3) или его несанкционированного использования.

Cross-site scripting (XSS) - название **атаки**.

В общем виде являются атакой на клиента, именно клиентские учетные данные атакующий может украсть.

- В некоторых случаях атака на систему, к примеру, если клиент администратор системы – атака на систему (деление атак условное и зависит от ситуации).

Соответствующий атаке **недостаток**: возможность внедрения **кода интерпретируемого на клиенте**.

Cross-site request forgery (CSRF) - название **атаки**.

http://docs.cntd.ru/document/1200102762

Терминология: уязвимости

ГОСТ Р ИСО/МЭК 27000-2012 Информационная технология. Методы и средства обеспечения безопасности. Системы менеджмента информационной безопасности. Общий обзор и терминология

2.46 **уязвимость** (vulnerability): Слабое место **актива** (2.3) или **меры и средства контроля и управления** (2.10), которое может быть использовано **угрозой** (2.45).

http://docs.cntd.ru/document/1200102762

Терминология: модель нарушителя

Модель нарушителя — (в информатике) абстрактное (формализованное или неформализованное) описание нарушителя правил разграничения доступа.

Модель нарушителя определяет:
• **категории** (типы) нарушителей, которые могут воздействовать на объект
• **цели**, которые могут преследовать нарушители каждой категории, возможный количественный состав, используемые инструменты, принадлежности, оснащение, оружие и проч.
• типовые **сценарии** возможных действий нарушителей, описывающие последовательность (алгоритм) и способы действий групп и отдельных нарушителей

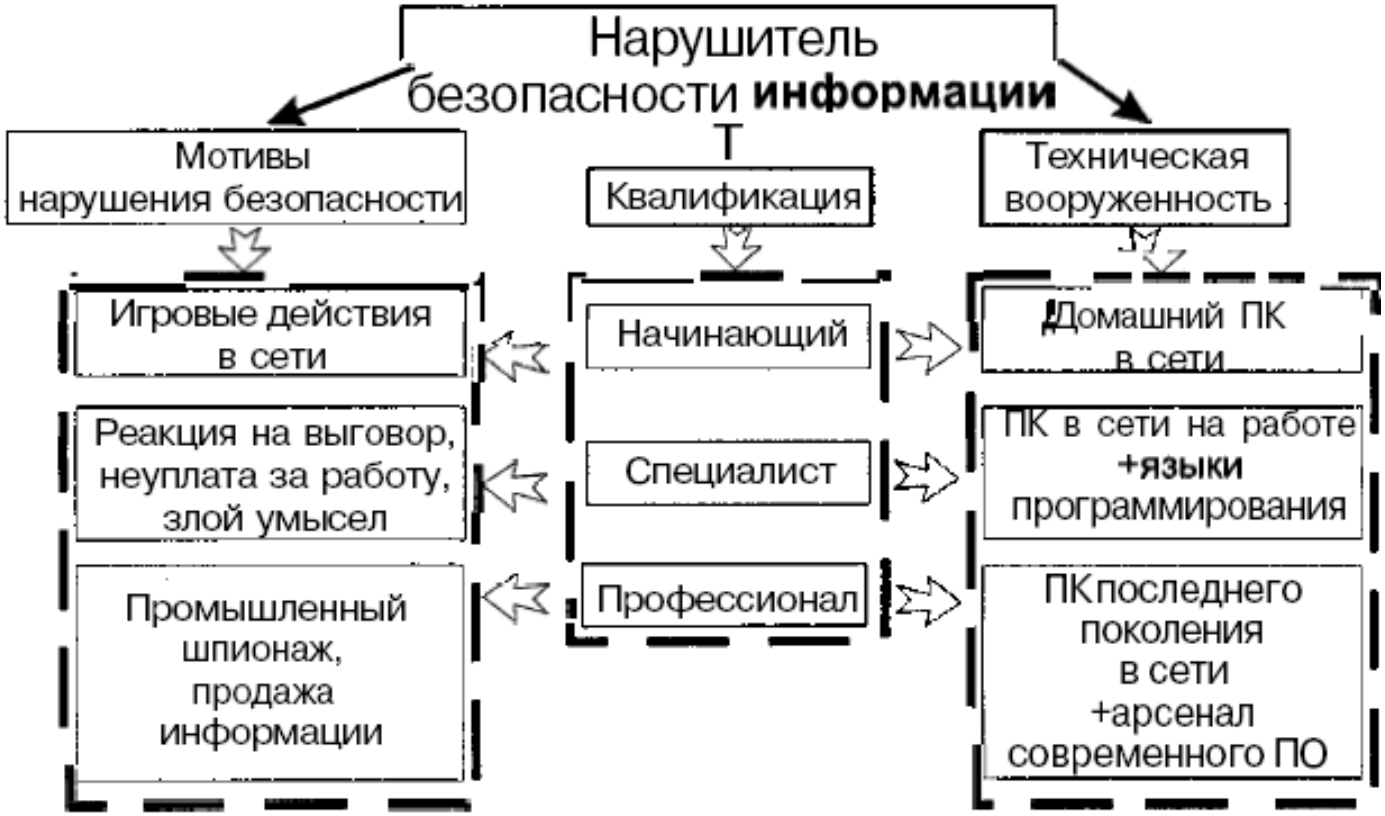
Модель нарушителей может иметь разную степень детализации.
• **Содержательная модель** нарушителей отражает систему принятых руководством объекта, ведомства взглядов на контингент потенциальных нарушителей, причины и мотивацию их действий, преследуемые цели и общий характер действий в процессе подготовки и совершения акций воздействия.
• **Сценарии воздействия** нарушителей определяют классифицированные типы совершаемых нарушителями акций с конкретизацией алгоритмов и этапов, а также способов действия на каждом этапе.
• **Математическая модель воздействия** нарушителей представляет собой формализованное описание сценариев в виде логико-алгоритмической последовательности действий нарушителей

Терминология: нарушители

С точки зрения наличия права постоянного или разового доступа в контролируемую зону нарушители могут подразделяться на два типа:

• нарушители, не имеющие права доступа в контролируемую зону территории (помещения) — **внешние нарушители**

• нарушители, имеющие право доступа в контролируемую зону территории (помещения) — **внутренние нарушители**



Терминология: защита

ГОСТ Р 50922-2006 Защита информации. Основные термины и определения

защита информации; ЗИ: Деятельность, направленная на **предотвращение утечки** защищаемой информации, несанкционированных и непреднамеренных **воздействий** на защищаемую информацию

• **правовая защита информации:** Защита информации правовыми методами, включающая в себя разработку законодательных и нормативных правовых документов (актов), регулирующих отношения субъектов по защите информации, применение этих документов (актов), а также надзор и контроль за их исполнением

• **техническая защита информации;** ТЗИ: Защита информации, заключающаяся в обеспечении некриптографическими методами безопасности информации (данных), подлежащей (подлежащих) защите в соответствии с действующим законодательством, с применением технических, программных и программно-технических средств

• **криптографическая защита информации:** Защита информации с помощью ее криптографического преобразования

• **физическая защита информации:** Защита информации путем применения организационных мероприятий и совокупности средств, создающих препятствия для проникновения или доступа неуполномоченных физических лиц к объекту защиты

ГОСТ Р 50922-2006

Терминология: парирование

ГОСТ Р 53114-2008 Защита информации. Обеспечение информационной безопасности в организации. Основные термины и определения

3.6.1 **обеспечение информационной безопасности** организации; обеспечение ИБ организации: Деятельность, направленная на **устранение (нейтрализацию, парирование)** внутренних и внешних **угроз** информационной безопасности организации или на **минимизацию ущерба** от возможной реализации таких угроз

3.1.15 **критически важная система информационной инфраструктуры;** *ключевая система информационной инфраструктуры*; КСИИ: Информационно-управляющая или информационно-телекоммуникационная система, которая осуществляет управление или информационное обеспечение критическим объектом или процессом, или используется для официального информирования общества и граждан, нарушение или прерывание функционирования которой (в результате деструктивных информационных воздействий, а также сбоев или отказов) может привести к чрезвычайной ситуации со значительными негативными последствиями

3.1.16 **критический объект: Объект** или **процесс**, нарушение непрерывности функционирования которого может нанести значительный ущерб.

http://docs.cntd.ru/document/1200075565

Терминология: критическая инфраструктура

Федеральный закон от 26 июля 2017 г. N 187-ФЗ "О безопасности критической информационной инфраструктуры Российской Федерации"

Статья 7. Категорирование объектов критической информационной инфраструктуры

1. Категорирование объекта критической информационной инфраструктуры представляет собой установление соответствия **объекта** критической информационной инфраструктуры критериям значимости и показателям их значений, присвоение ему одной из **категорий значимости**, проверку сведений о результатах ее присвоения.

3. Устанавливаются **три категории** значимости объектов критической информационной инфраструктуры - первая, вторая и третья

4. **Субъекты** критической информационной инфраструктуры присваивают одну из категорий значимости **принадлежащим** им на праве **собственности**, **аренды** или ином законном основании **объектам** критической информационной инфраструктуры. Если объект критической информационной инфраструктуры не соответствует критериям значимости, показателям этих критериев и их значениям, ему не присваивается ни одна из таких категорий

187-ФЗ

Терминология: критическая инфраструктура

Федеральный закон от 26 июля 2017 г. N 187-ФЗ "О безопасности критической информационной инфраструктуры Российской Федерации"

2. Категорирование осуществляется исходя из:
1) **социальной** значимости, выражающейся в оценке возможного ущерба, причиняемого жизни или здоровью людей, возможности прекращения или нарушения функционирования объектов обеспечения жизнедеятельности населения, транспортной инфраструктуры, сетей связи, а также максимальном времени отсутствия доступа к государственной услуге для получателей такой услуги;
2) **политической** значимости, выражающейся в оценке возможного причинения ущерба интересам Российской Федерации в вопросах внутренней и внешней политики;
3) **экономической** значимости, выражающейся в оценке возможного причинения прямого и косвенного ущерба субъектам критической информационной инфраструктуры и (или) бюджетам Российской Федерации;
4) **экологической** значимости, выражающейся в оценке уровня воздействия на окружающую среду;
5) значимости объекта критической информационной инфраструктуры для обеспечения **обороны** страны, **безопасности** государства и **правопорядка**.

187-ФЗ УК РФ Статья 274.1

Модно: Agile

Гибкая методология разработки (англ. *Agile software development*, *agile-методы*) — серия подходов к разработке программного обеспечения, ориентированных на использование итеративной разработки, динамическое формирование требований и обеспечение их реализации в результате постоянного взаимодействия внутри самоорганизующихся рабочих групп, состоящих из специалистов различного профиля

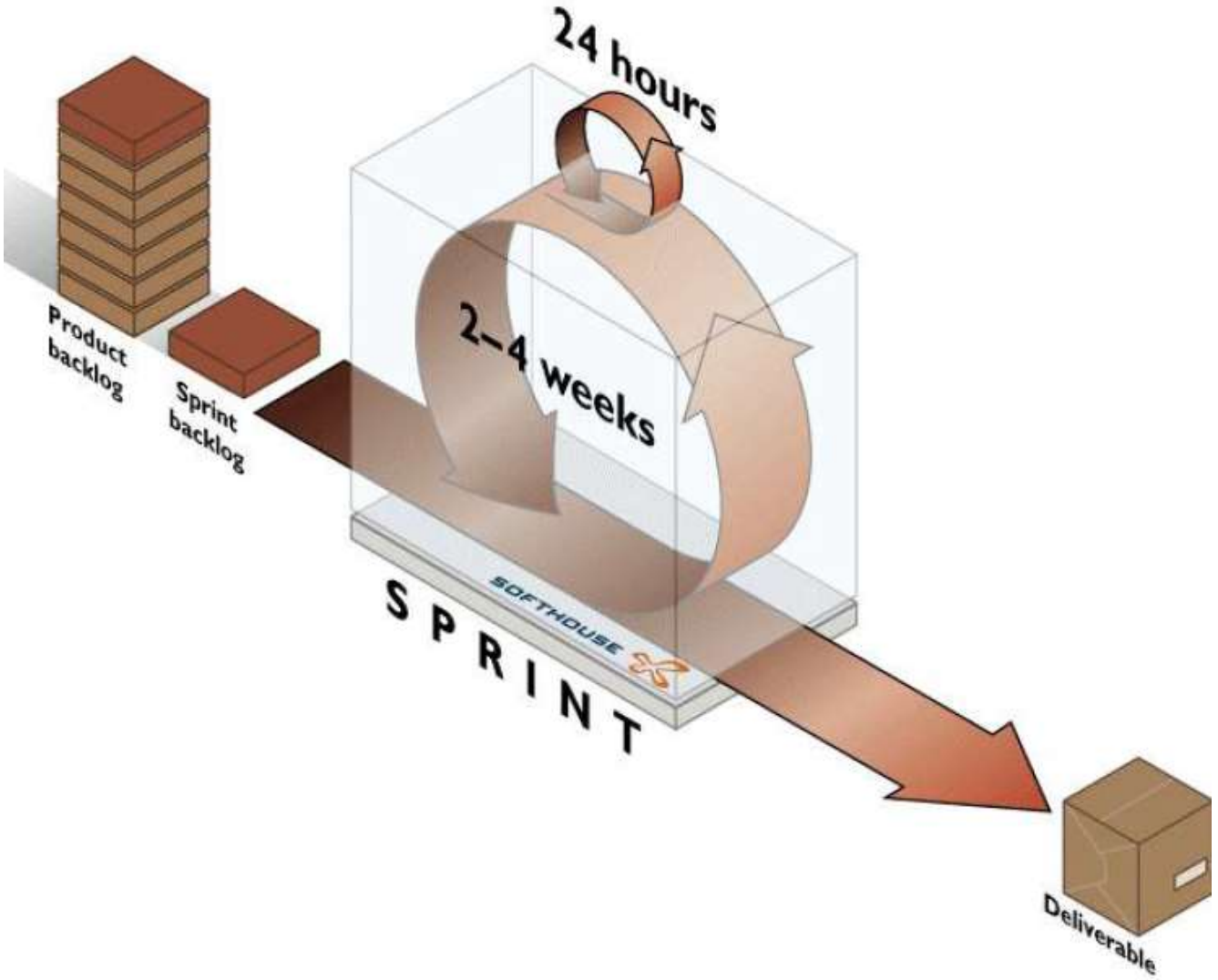
Существует несколько методик, относящихся к классу гибких методологий разработки, в частности экстремальное программирование, DSDM, Scrum, FDD.

- Люди и взаимодействия важнее чем процессы и инструменты
- Работающий код важнее совершенной документации
- Сотрудничество с заказчиком важнее контрактных обязательств
- Реакция на изменения важнее следования плану

Манифест (2001) Agile

Модно: Scrum

Scrum (/skrʌm/; англ. *scrum* «схватка») — методология гибкой разработки ПО. Методология делает акцент на **качественном контроле** процесса разработки.



Scrum (1986)
Agile

Модно: Scrum

Журнал пожеланий проекта

Журнал пожеланий проекта (англ. *Project backlog*) — это **список требований** к функциональности, упорядоченный по их степени важности, подлежащих реализации

Элементы этого списка называются пользовательскими историями (*user story*) или элементами беклога (*backlog items*). Журнал пожеланий проекта **открыт** для редактирования **для всех участников** скрам-процесса. Project backlog ведется SCRUM Product Owner

Журнал пожеланий спринта

Журнал пожеланий спринта (англ. *Sprint backlog*) — содержит функциональность, выбранную владельцем продукта из журнала пожеланий проекта

Все **функции разбиты по задачам**, каждая из которых **оценивается** скрам-командой. На Sprint Planning Meeting команда оценивает объем работы, который нужно проделать для завершения спринта методом Planning Poker

Scrum (1986)
Agile

Модно: Scrum

Спринт — **итерация** в скраме, в ходе которой создается инкремент бизнес-продукта. **Жестко фиксирован по времени**. Длительность одного спринта от 1 до 4 недель

Возможности к реализации в очередном спринте определяются в начале спринта на совещании **Sprint Planning Meeting** планирования методом **Planning Poker** и **не могут изменяться** на всем его протяжении. При этом строго фиксированная небольшая длительность спринта придает процессу разработки предсказуемость и гибкость

Чем короче спринт, тем более гибким является процесс разработки, **релизы** выходят чаще, быстрее поступают отзывы от потребителя, меньше времени тратится на работу в неправильном направлении

С другой стороны, при более длительных спринтах скрам-команда уменьшает издержки на совещания, демонстрации продукта и т. п.

Для оценки объема работ в спринте можно использовать предварительную оценку, измеряемую в **очках истории**. Предварительная **оценка длины** спринта фиксируется в бэклоге проекта.

Scrum (1986)
Agile

Модно: CI

Непрерывная интеграция (*CI*, англ. *Continuous Integration*) — практика разработки программного обеспечения, которая заключается в **постоянном слиянии** рабочих копий в общую основную ветвь разработки (до нескольких раз в день) и выполнении частых **автоматизированных сборок** проекта для скорейшего выявления потенциальных дефектов и решения интеграционных проблем

В обычном проекте, где над разными частями системы разработчики трудятся независимо, стадия интеграции является заключительной. Она может непредсказуемо задержать окончание работ

Переход к непрерывной интеграции позволяет снизить трудоёмкость интеграции и сделать её более предсказуемой за счёт наиболее раннего обнаружения и устранения ошибок и противоречий, но основным преимуществом является сокращение **стоимости исправления дефекта**, за счёт **раннего его выявления**

Сборка по расписанию
Agile

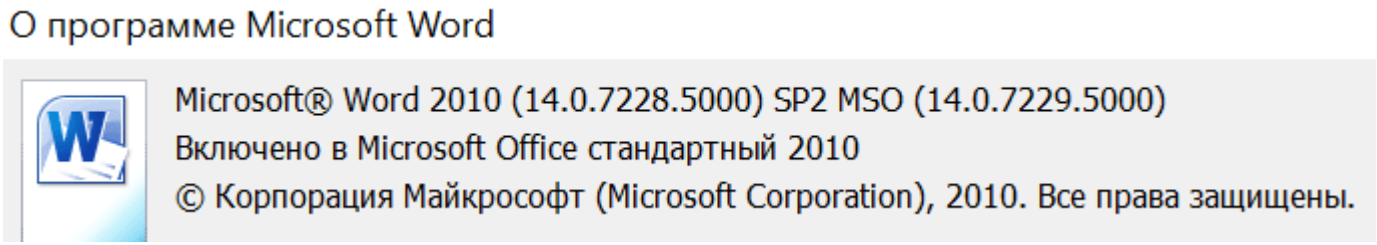
Модно: CI

Сборки по расписанию (англ. *daily **build** — ежедневная сборка*), как правило, проводятся в нерабочее время, ночью (англ. *nightly build*), планируются таким образом, чтобы **к началу очередного рабочего дня** были готовы **результаты тестирования**

Для различия дополнительно вводится система нумерации сборок — обычно, **каждая сборка нумеруется** натуральным числом, которое **увеличивается с каждой новой сборкой**

Исходные тексты и другие исходные данные при взятии их из репозитория (хранилища) системы контроля версий помечаются **номером сборки**

Благодаря этому, точно такая же сборка может быть **точно воспроизведена в будущем** — достаточно взять исходные данные по нужной метке и запустить процесс снова. Это даёт возможность повторно выпускать даже очень старые версии программы с небольшими исправлениями.



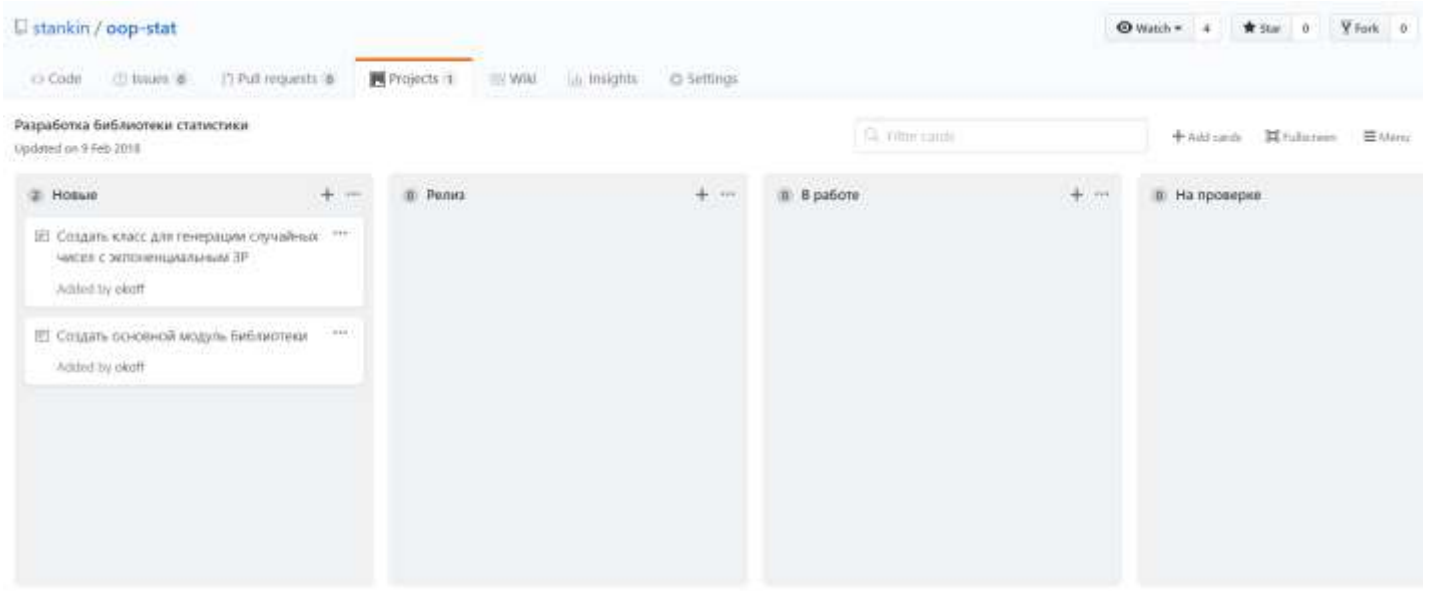
Сборка по расписанию
Agile

Средства: репозиторий

Репозиторий, хранилище — место, где хранятся и поддерживаются какие-либо данные. Чаще всего данные в репозитории хранятся в виде файлов, доступных для дальнейшего распространения по сети

Репозитории используются в системах управления версиями, в них хранятся все документы вместе с историей их изменения и другой служебной информацией

Система управления версиями (от англ. *Version Control System, VCS* или *Revision Control System*) — программное обеспечение для облегчения работы с изменяющейся информацией. Система управления версиями позволяет хранить несколько версий одного и того же документа, при необходимости возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение, и многое другое



Репозиторий

Средства: инструменты DevOps

DevOps — это командная работа сотрудников, занимающимися **разработкой**, **операциями** и **тестированием**, а также необходимый набор инструментов

Как правило, **инструменты** DevOps вписываются в одну или несколько из этих категорий, что отражает ключевые аспекты **разработки** и **доставки** программного обеспечения:

- **Code** — разработка и анализ кода, инструменты контроля версий, слияние кода
- **Build** — инструменты непрерывной интеграции, статус сборки
- **Test** — инструменты непрерывного тестирования, которые обеспечивают обратную связь по бизнес-рискам
- **Package** — репозиторий артефактов, предварительная установка приложения
- **Release** — управление изменениями, официальное утверждение выпуска, автоматизация выпуска
- **Configure** — Конфигурация и управление инфраструктурой, Инфраструктура как инструменты кода
- **Monitor** — мониторинг производительности приложений, опыт работы с конечным пользователем

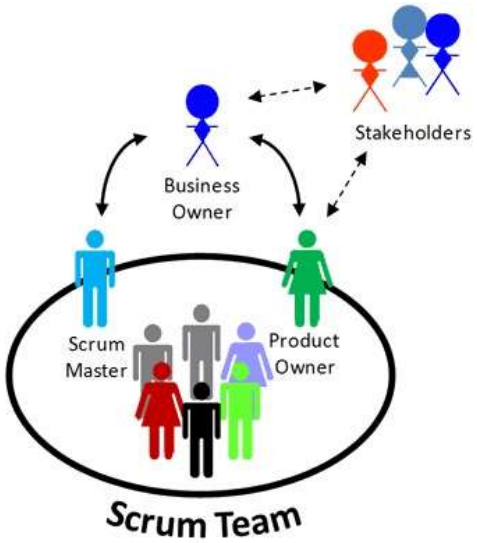
DevOps (2009)

Модно: Scrum

Задачи истории спринта (Sprint Story Tasks)
Добавляются к историям спринта. Выполнение каждой задачи оценивается в часах. Каждая задача не должна превышать 12 часов (зачастую команда настаивает, чтобы максимальная продолжительность задачи равнялась одному рабочему дню)

Ежедневное стоячее SCRUM-совещание (Daily SCRUM)

- начинается в одно и то же время в одном месте
- все могут наблюдать, но только «свиньи» говорят
- в митинге участвуют SCRUM Master, SCRUM Product Owner и SCRUM Team
- **длится ровно 15 минут**
- все участники во время Daily SCRUM стоят (митинг в формате Daily Standup)



[Scrum (1986)]
[Agile]

Модно: Scrum

Ежедневное стоячее SCRUM-совещание (Daily SCRUM)

SCRUM-мастер задает каждому члену SCRUM-команды три вопроса:

- что я **сделал** с момента прошлой встречи для того, чтобы помочь команде разработки достигнуть цели спринта?
- что я **сделаю** сегодня для того, чтобы помочь команде разработки достичь цели спринта?
- вижу ли я **препятствия** для себя или команды разработки, которые могли бы затруднить достижение цели спринта?

Над решением этих проблем методом фасилитации работает скрам-мастер. Обычно это решение проходит за рамками ежедневного совещания и в составе лиц, непосредственно затронутых данным препятствием.

Scrum (1986)
Agile

Модно: Scrum

Обзор итогов спринта (Sprint review meeting)

Проводится **в конце спринта.**

- Команда демонстрирует прирост инкремента продукта всем заинтересованным лицам.
- Все члены команды участвуют в демонстрации (один человек на демонстрацию или каждый показывает, что сделал за спринт).
- Нельзя демонстрировать незавершенную функциональность.
- Ограничена четырьмя часами в зависимости от продолжительности итерации и прироста функциональности продукта.

Груминг беклога (Grooming)

Беклог отправляется в парикмахерскую для того, чтобы скрам-команда и владелец продукта могли:

- Добавить, убрать или разбить элементы беклога продукта (PBI).
- Уточнить или дать новые оценки.
- Изменить порядок следования элементов беклога продукта.
- Обсудить и прояснить требования.

Scrum (1986)
Agile

Модно: Scrum

Ретроспективное совещание (Retrospective meeting)
Проводится в конце спринта.

Члены скрам-команды, скрам-мастер и продукт-оунер высказывают свое мнение о прошедшем спринте.

Скрам-мастер задает два вопроса всем членам команды:

Что было сделано хорошо в прошедшем спринте?
Что надо улучшить в следующем?

Выполняют улучшение процесса разработки (обсуждают варианты решения проблем, фиксируют удачные решения и вызвавшегося владельца продукта).

Ограничена четырьмя часами для спринта любой длины.

Scrum (1986)
Agile

Модно: Scrum

Ретроспективное совещание (Retrospective meeting)
Проводится в конце спринта.

Члены скрам-команды, скрам-мастер и продукт-оунер высказывают свое мнение о прошедшем спринте.

Скрам-мастер задает два вопроса всем членам команды:

Что было сделано хорошо в прошедшем спринте?
Что надо улучшить в следующем?

Выполняют улучшение процесса разработки (обсуждают варианты решения проблем, фиксируют удачные решения и вызвавшегося владельца продукта).

Ограничена четырьмя часами для спринта любой длины.

Scrum (1986)
Agile

Терминология: проблема и обходной путь

Проблема (Problem) – неизвестная **причина** одного или более инцидентов

Во время создания записи о проблеме причина инцидента обычно неизвестна и подсистема помогает менеджеру по управлению проблемами осуществлять исследование и поиск возможных причин.

В **управлении проблемами (Problem Management)** используется база данных **известных ошибок (Known Error)** – это проблемы, у которых документированы их **корневые причины** и **обходной путь**

Обходной путь (Workaround) – **снижение** или **устранение влияния** инцидентов или проблем, для которых еще **невозможно их полное разрешение**

Запла́тка или **патч** (англ. *patch* /pætʃ/ — заплатка) — информация, предназначенная для автоматизированного внесения определённых изменений в компьютерные файлы. Патчем или *обновлением* англ. *update* называется, в частности, содержащее такую информацию автоматизированное отдельно поставляемое программное средство, используемое для устранения проблем в программном обеспечении или изменения его функциональности

Патч

Лекция 12 «**Управление конфигурацией**»

Овчинников П.Е.
МГТУ «СТАНКИН»,
ст.преподаватель кафедры ИС

06.02.2019

Конфигурация = документированность

Модель процессов

Модель архитектуры ИС

Модель данных ИС

Терминология: инженерия

ГОСТ Р 57193-2016 Системная и программная инженерия. Процессы жизненного цикла систем

системная инженерия (systems engineering): междисциплинарный подход, управляющий **полным техническим и организаторским усилием**, требуемым для преобразования ряда потребностей заинтересованных сторон, ожиданий и ограничений в решение и для поддержки этого решения в течение его жизни

ISO/IEC/IEEE 24765:2017 Systems and software engineering -- Vocabulary

программная инженерия (англ. *software engineering*): приложение

- **систематического**
- **дисциплинированного**
- **измеримого**

подхода к

- **разработке**
- **функционированию и**
- **сопровождению**

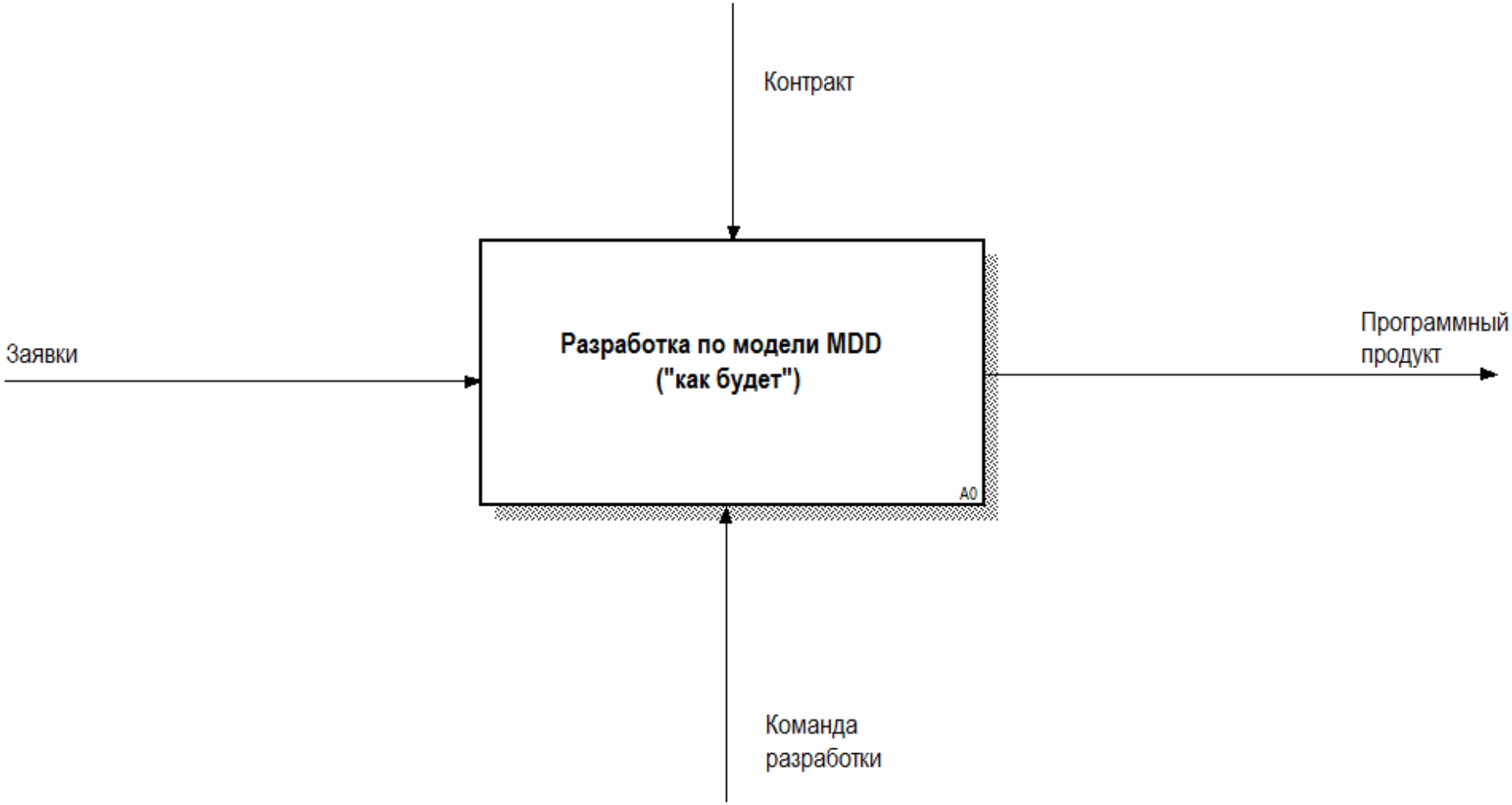
программного обеспечения,

а также к исследованию этих подходов;

то есть, приложение дисциплины инженерии к программному обеспечению

ГОСТ Р 57193-2016

Инженерия в модели MDD



Гибкая разработка: модель MDD

Разработка, управляемая моделями (*model-driven development,* **MDD**, *Model-driven engineering*, **MDE**) — это **стиль разработки** программного обеспечения, когда **модели** становятся **основными артефактами** разработки, из которых **генерируется код** и **другие артефакты**

Модель — это **абстрактное описание** программного обеспечения, которое скрывает информацию о некоторых аспектах с целью представления упрощенного описания остальных

Модель может быть **исходным артефактом** в разработке, если она фиксирует информацию в **форме**, пригодной для **интерпретаций людьми** и обработки **инструментальными средствами**

В разработке, управляемой моделями, конфигурация программного и информационного обеспечения всегда точно описана в базах данных, описывающих модели!

MDE

Гибкая разработка: модель MDD

Модель определяет **нотацию** и **метамодель**

Нотация представляет собой совокупность **графических элементов**, которые применяются в модели и могут быть интерпретированы людьми

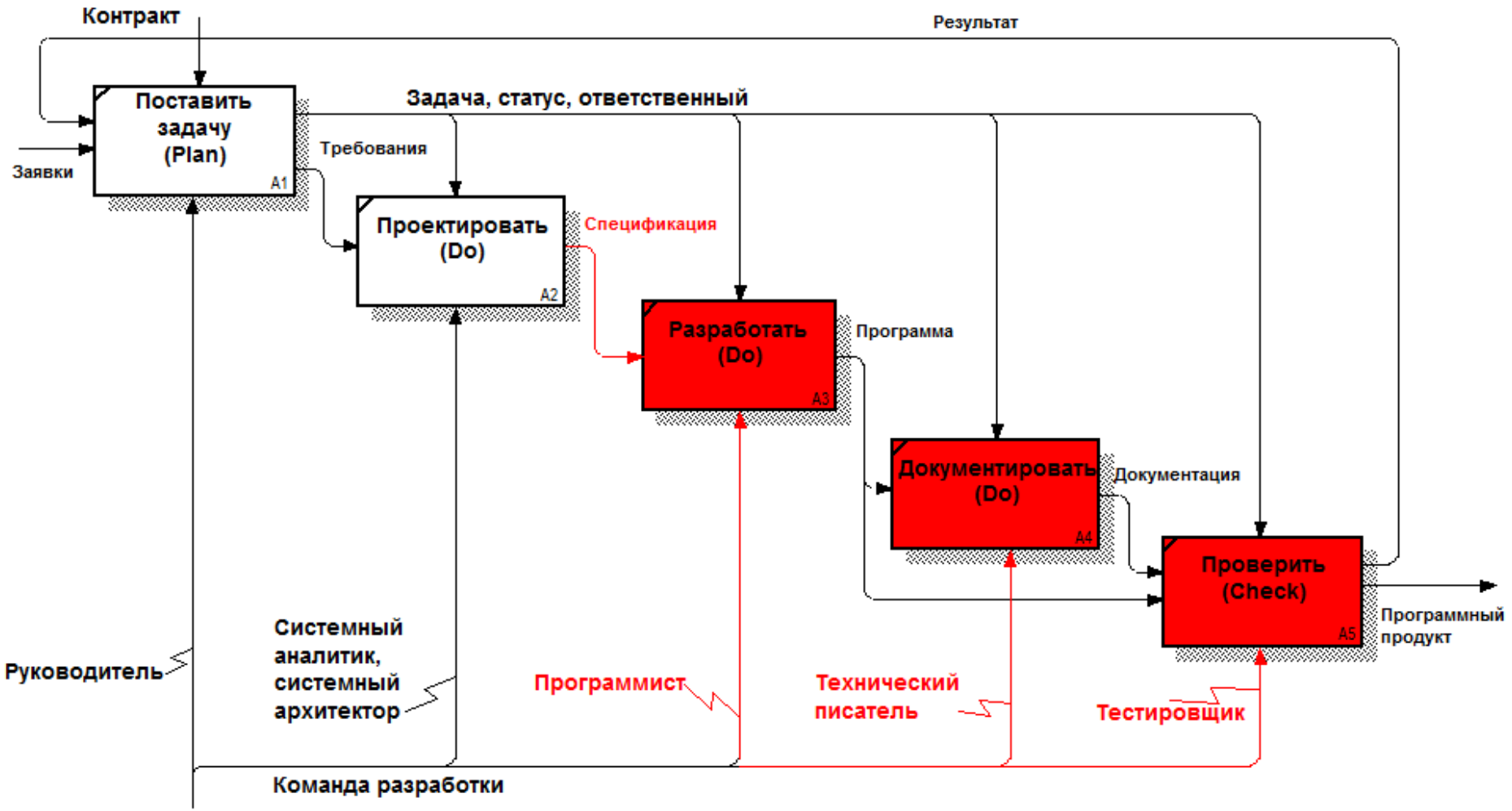
Метамодель описывает используемые в модели понятия и фиксирует информацию в виде метаданных, которые могут быть обработаны инструментальными средствами.

Наиболее известными современными MDE-инициативами являются:
1.разработка Object Management Group (OMG) под названием model-driven architecture (MDA)
2.экосистема Eclipse для инструментов моделирования и программирования (Eclipse Modeling Framework)

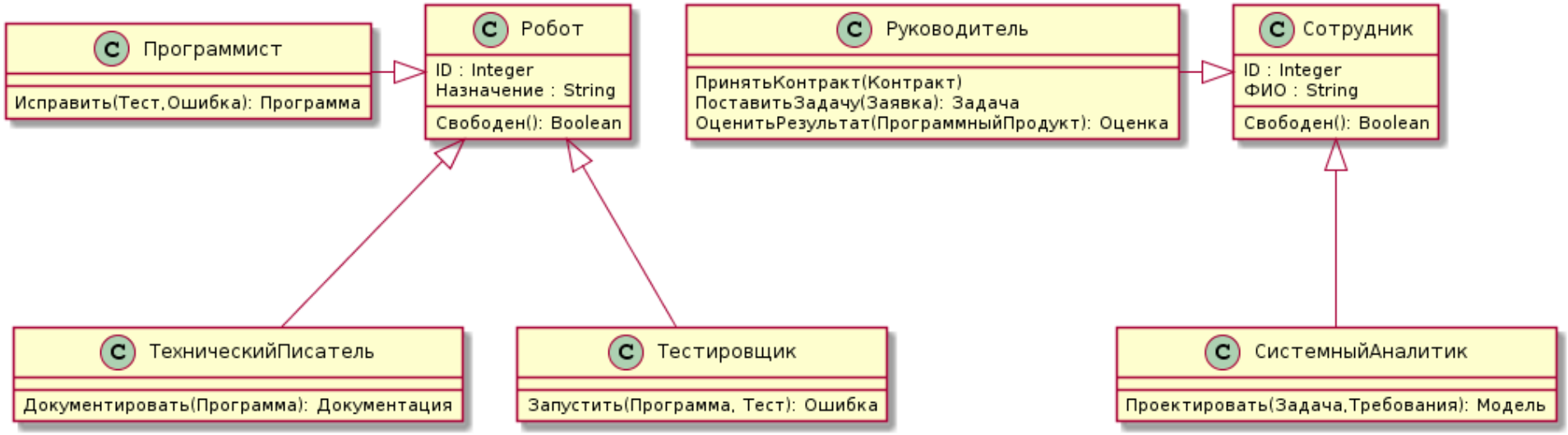
В отечественной ИТ-индустрии наиболее известным продуктом, поддерживающим MDD, является платформа «1С:Предприятие»

Объект проектирования: метамодель

Модель MDD и команда



Модель MDD и команда



http://www.intuit.ru/studies/courses/3467/709/lecture/21825
http://www.intuit.ru/studies/courses/3467/709/lecture/21825?page=2

Модель MDD и команда

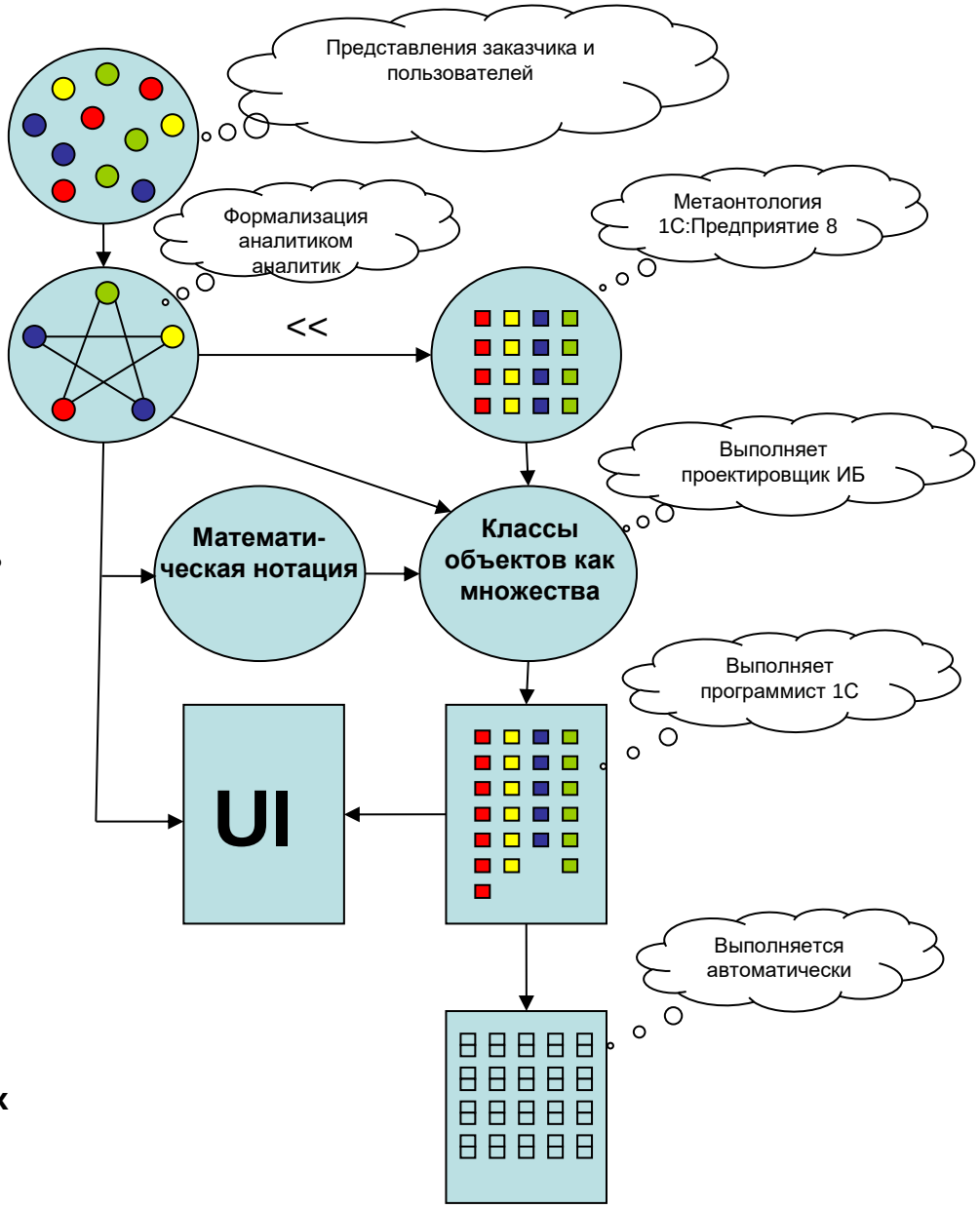
1. **Семантическая сеть**

2. **Онтология**

3. **Концептуальная модель**

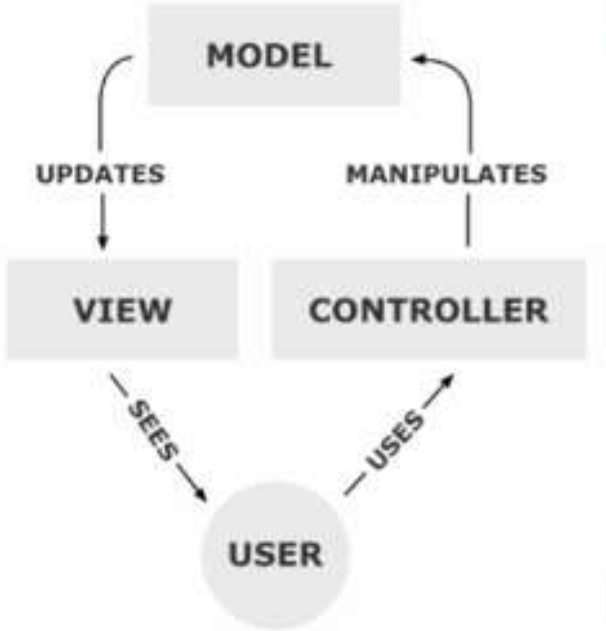
4. **Объектная модель информационной базы 1С:Предприятие**

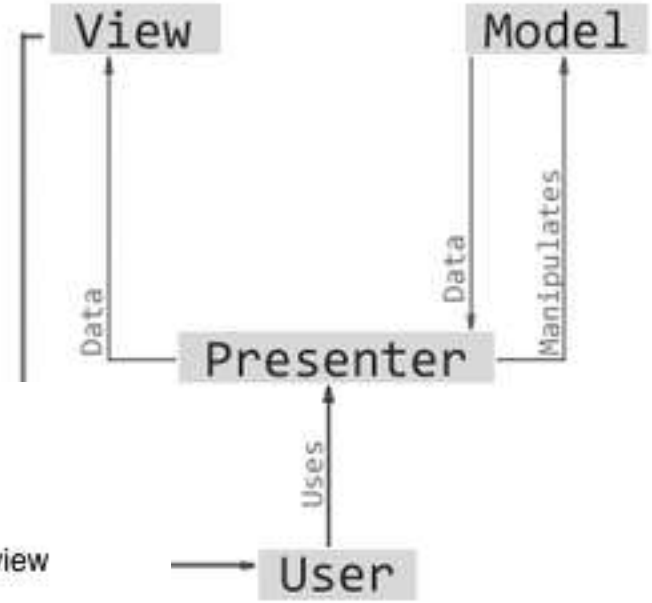
5. **Логическая модель реляционной базы данных**

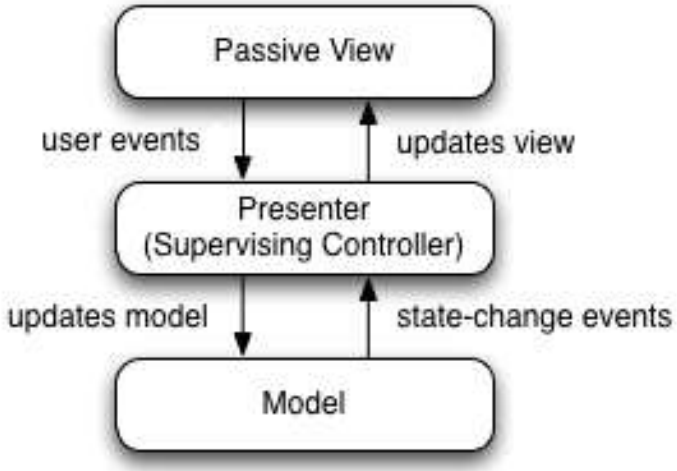


Частичная реализация принципов MDD

Паттерны MVC



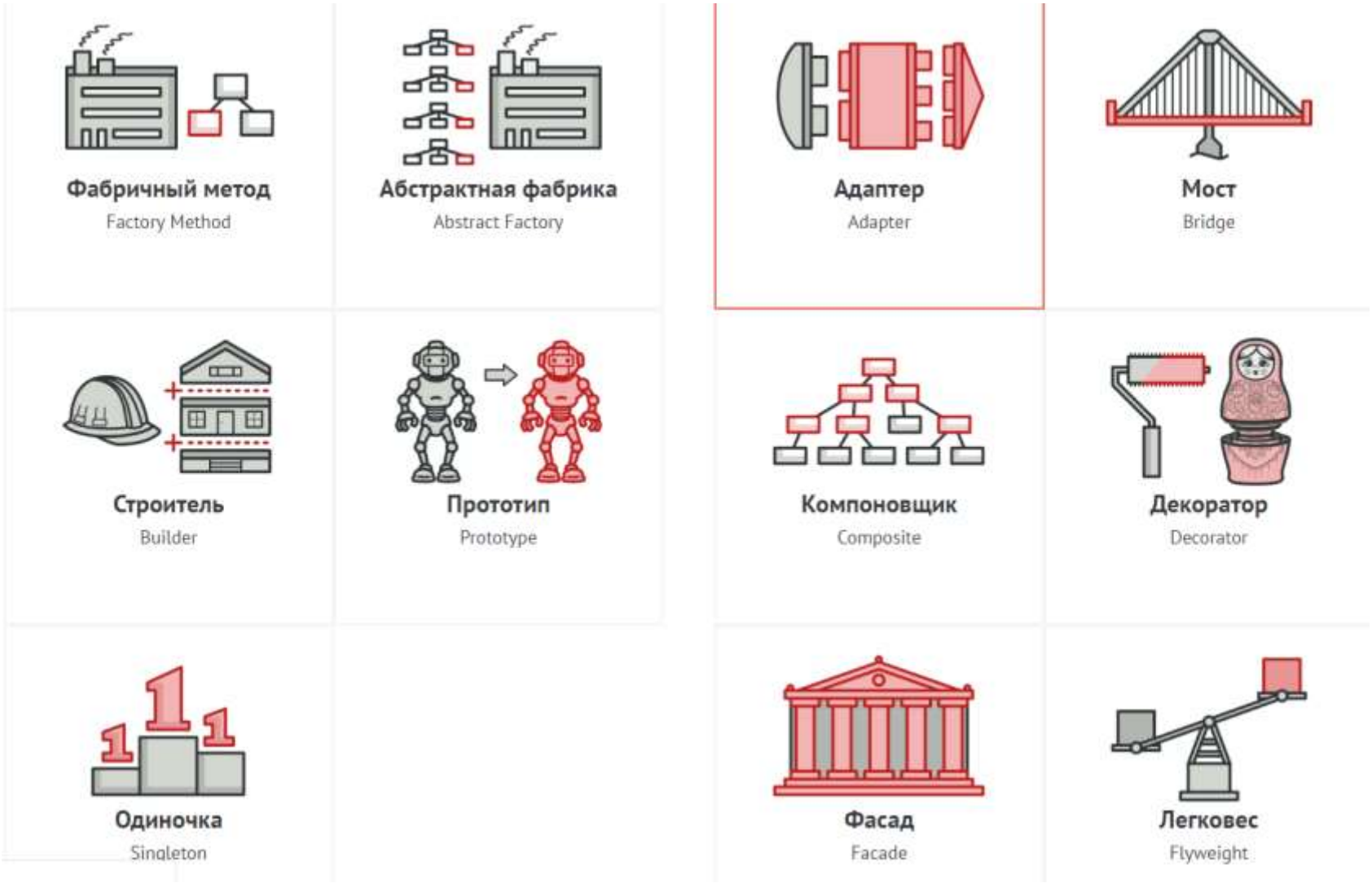




[Model-View-Controller]

Частичная реализация принципов MDD

Паттерны ООП



Архитектурные паттерны

Частичная реализация принципов MDD

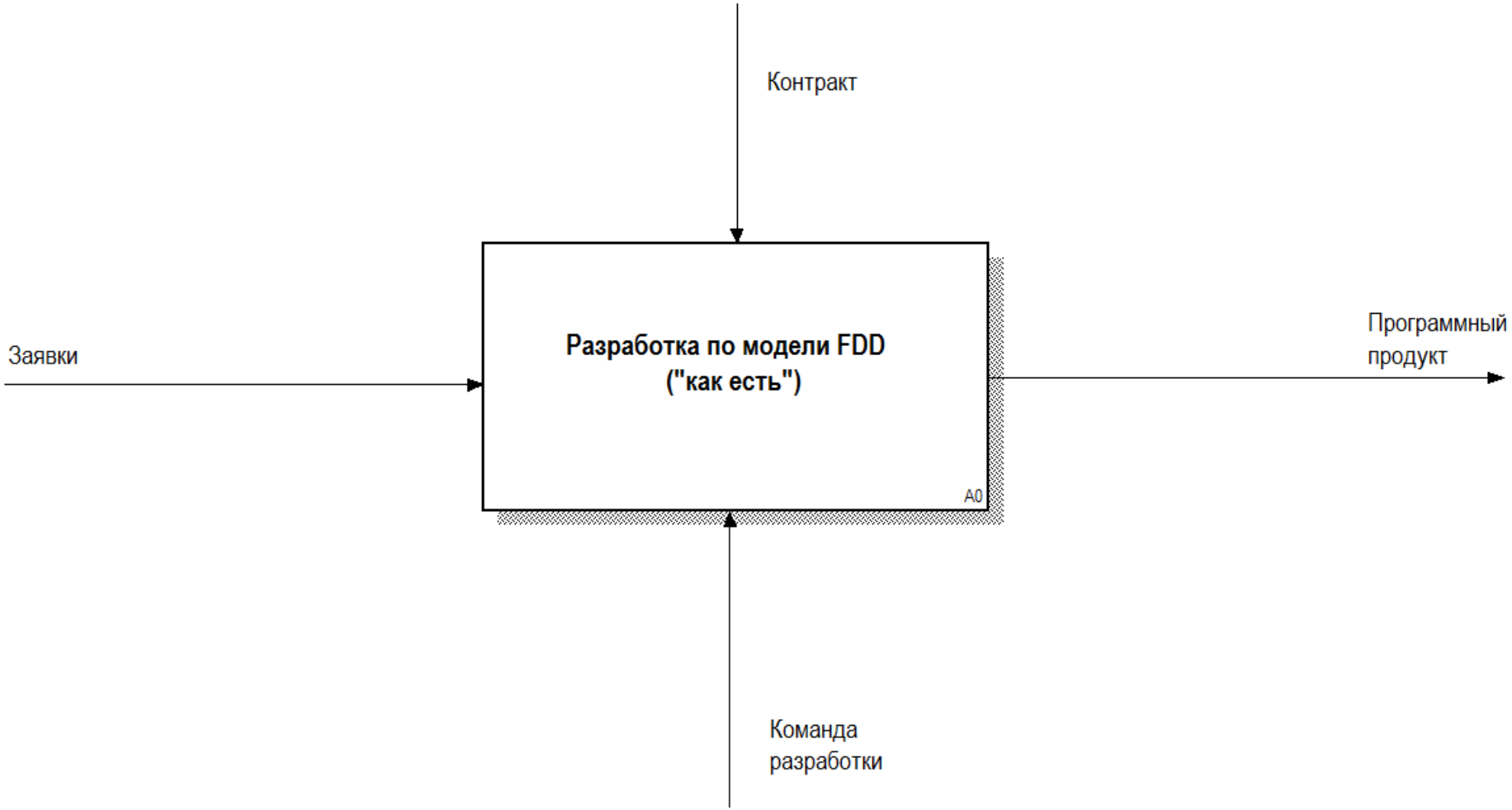
Постпроцессоры

PostCSS — программа, которая автоматизирует рутинные операции с CSS с помощью расширений, написанных на языке JavaScript

Используется при разработке Википедии, Facebook и GitHub

Один из самых часто загружаемых с npm инструментов для работы с CSS

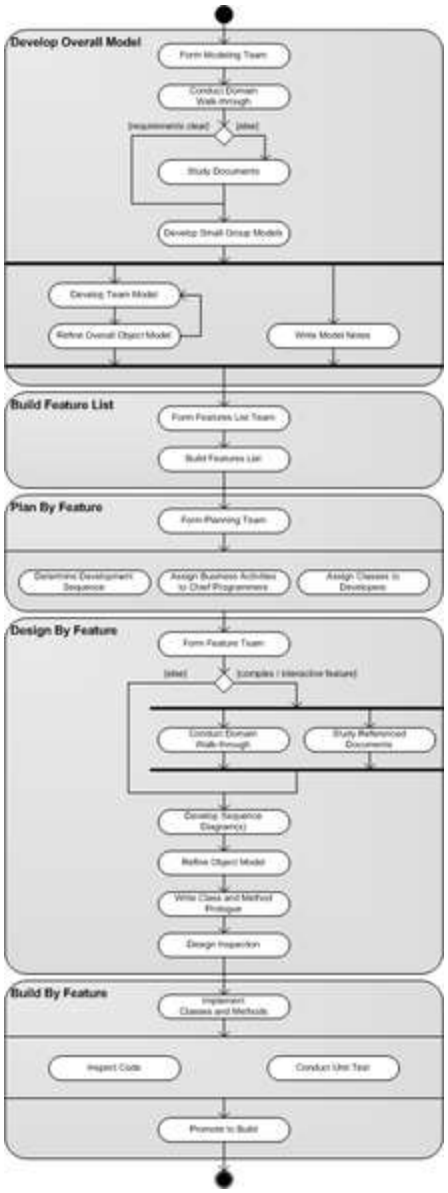
Инжиниринг в модели разработки FDD



Гибкая разработка: модель FDD

Feature driven development (**FDD**, *разработка, управляемая функциональностью*) — итеративная методология разработки программного обеспечения, одна из *гибких методологий разработки (agile)*

FDD представляет собой попытку объединить наиболее признанные в индустрии разработки программного обеспечения методики, принимающие за основу важную для заказчика функциональность (свойства) разрабатываемого программного обеспечения. Основной целью данной методологии является разработка реального, работающего программного обеспечения систематически, в поставленные сроки.

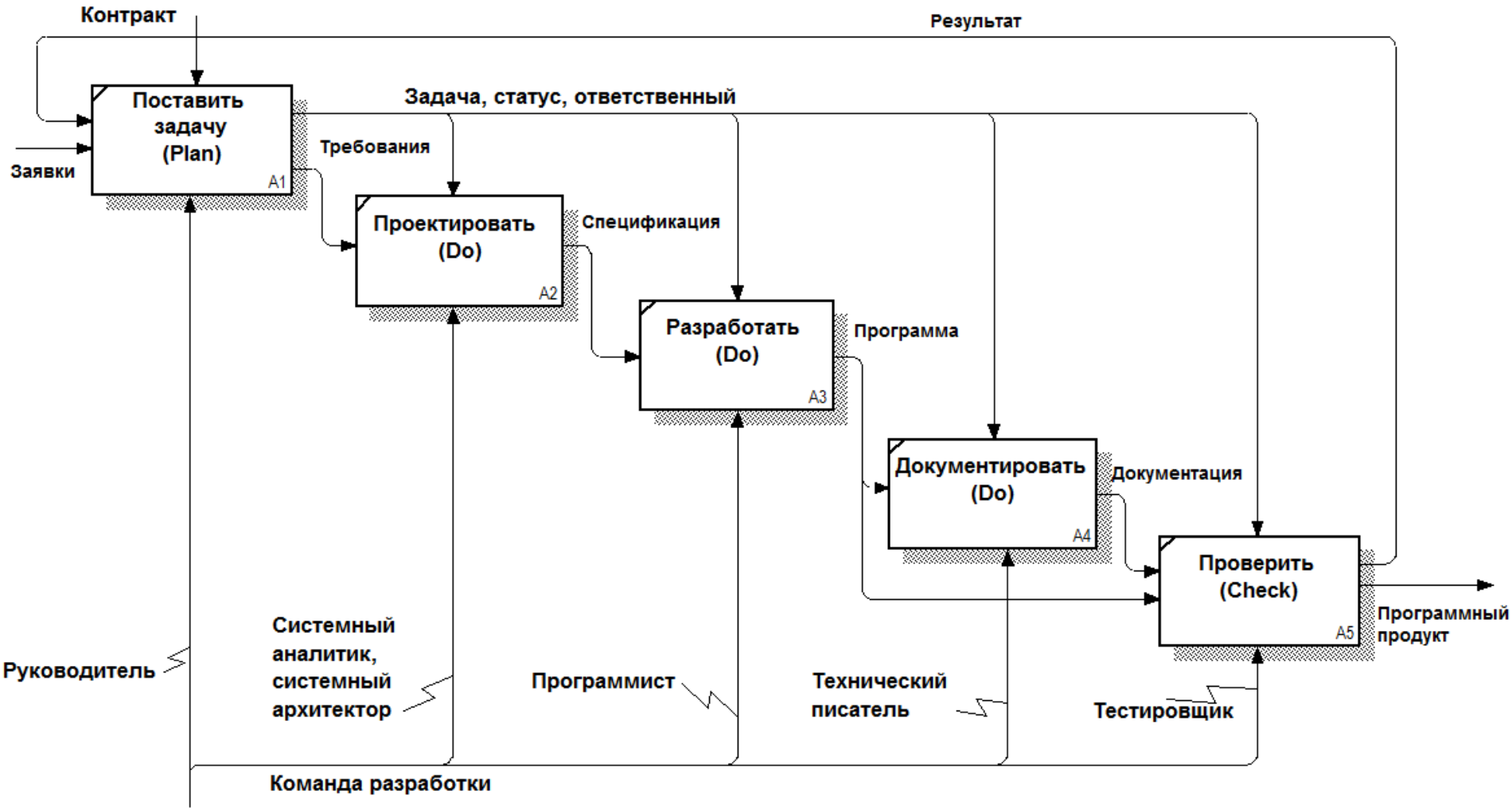


FDD включает в себя пять базовых видов деятельности:

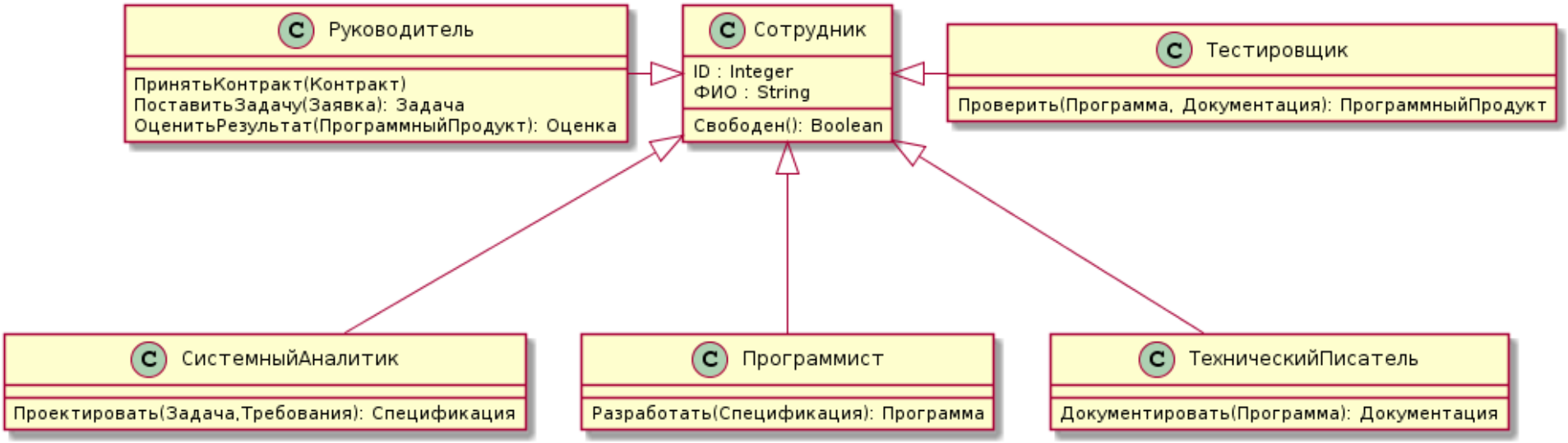
1. разработка **общей модели**
2. составление **списка необходимых функций** системы
3. **планирование** работы над каждой функцией
4. **проектирование** функции
5. **реализация** функции

https://ru.wikipedia.org/wiki/Feature_driven_development
http://www.intuit.ru/studies/courses/3505/747/lecture/26289?page=1#sect3
http://www.intuit.ru/studies/courses/2188/174/lecture/4730?page=2

Модель FDD и команда



Модель FDD и команда



http://www.intuit.ru/studies/courses/3467/709/lecture/21825
http://www.intuit.ru/studies/courses/3467/709/lecture/21825?page=2

Реинжиниринг

Обра́тная разрабо́тка (**обратное проектирование, обратный инжиниринг**, **реверс-инжиниринг**; англ. *reverse engineering*) — исследование некоторого готового устройства или программы, а также документации на него с целью понять принцип его работы; например, чтобы

- обнаружить недокументированные возможности (в том числе программные закладки),
- сделать изменение или
- воспроизвести устройство, программу или иной объект с аналогичными функциями, но без прямого копирования

Применяется обычно в том случае, если **создатель** оригинального объекта **не предоставил** информации о **структуре** и **способе создания** (производства) объекта

Правообладатели таких объектов могут заявить, что проведение **обратной разработки** или использование её результатов нарушает их исключительное право по закону об авторском праве и патентному законодательству

Реинжиниринг

Реинжиниринг программного обеспечения — процесс создания новой функциональности или устранения ошибок, путём революционного изменения, но используя уже имеющееся в эксплуатации программное обеспечение

Как правило, утверждается, что «легче разработать новый программный продукт». Это связано со следующими проблемами:

- реинжиниринг, чаще всего, дороже разработки нового программного обеспечения, так как требуется убрать ограничения предыдущих версий, при этом сохранив с ними совместимость
- реинжиниринг не может сделать программист низкой и средней квалификации — даже профессионалы часто не могут качественно реализовать его, поэтому требуется работа программистов с большим опытом переделки программ и знанием различных технологий
- разработчику бывает сложно разбираться в чужом исходном коде — это вынуждает адаптироваться к восприятию незнакомого стиля программирования, расходует время на всесторонний анализ и освоение реализованных в проекте концепций, используемых в нём сторонних библиотек, требует скрупулёзно исследовать принцип действия всех плохо документированных участков кода — и всё это лишь осложняет процесс перехода продукта на новые архитектурные решения
- кроме того, сам характер деятельности требует дополнительной мотивации: по сравнению с созданием новых продуктов, переработка уже имеющихся не всегда приносит столь же наглядные и впечатляющие результаты, зачастую отягощает грузом технического долга и оставляет мало места для профессионального самовыражения.

Реинжиниринг в FDD

Гарантированно есть список **решенных задач** и **возможно** есть **набор тестов**, связанных или не связанных с задачами

Шаг 1. Восстановление требований
Вход: **задачи**
Выход: пользовательские **истории**

Эпик, тема, история (мнение)

Реинжиниринг в TFD/TDD/BDD

Гарантированно есть список **решенных задач** и **точно** есть **набор тестов**, связанных с задачами

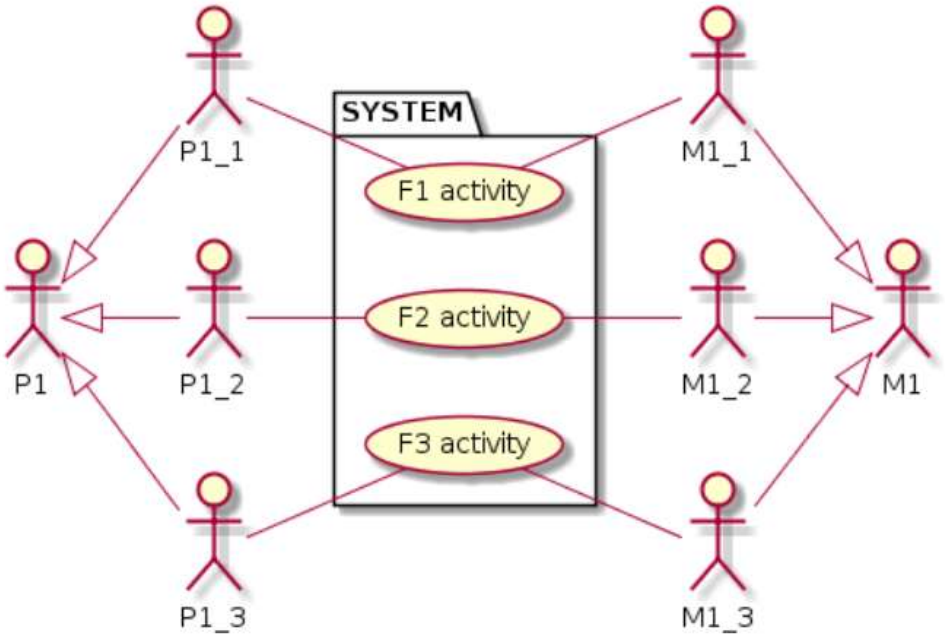
Шаг 1. Восстановление требований
Вход: **тесты**
Выход: пользовательские **истории**

Эпик, тема, история (мнение)

Реинжиниринг в TFD/TDD/BDD

Гарантированно есть список **решенных задач** и **точно** есть **набор тестов**, связанных с задачами

Шаг 2. Восстановление прецедентов
Вход: **истории**
Выход: **прецеденты**



Эпик, тема, история (мнение)

Прецеденты: IDEF0 → Use Case

Описание решения

Общее решение состоит в следующей ассоциации элементов диаграммы IDEF0 с элементами диаграммы прецедентов:

- стрелки механизмов преобразуются в "actor";
- декомпозируемые механизмы становятся родительскими "actor";
- имена блоков становятся именами прецедентов;
- все блоки дочерней диаграммы объединяются в один пакет с именем родительской.

Особенности преобразования

При преобразовании диаграмм IDEF0 в диаграммы прецедентов UML **теряется** информация **обо всех** информационных и материальных потоках - о входах, выходах и управлении

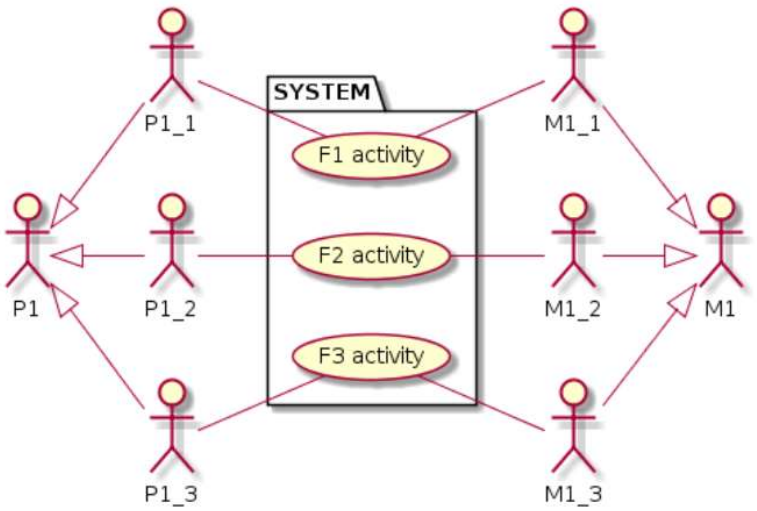
При обратном преобразовании все эти потоки должны быть **восстановлены** или **спроектированы** заново

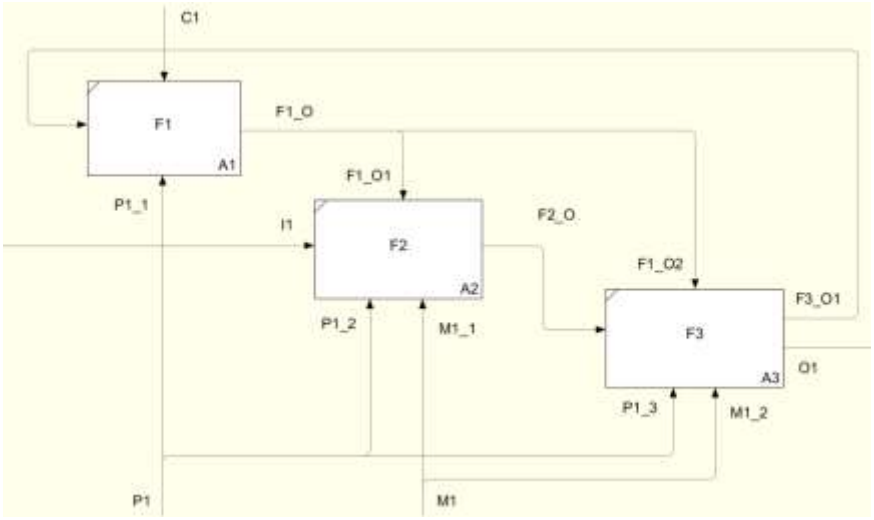
Реинжиниринг в FDD/TFD/TDD/BDD

Шаг 3. Восстановление функциональной структуры

Вход: **прецеденты**

Выход: **модель процессов**

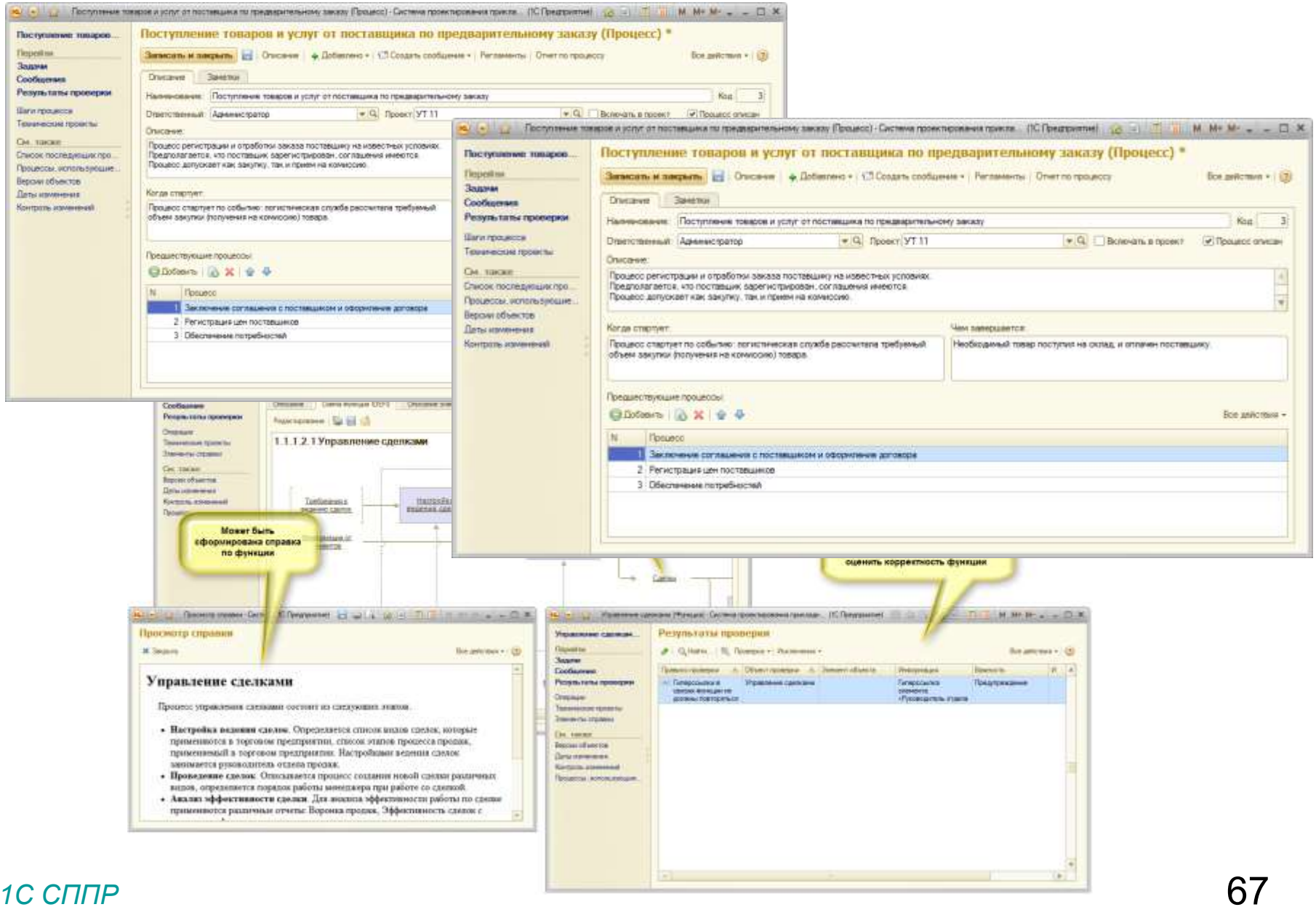




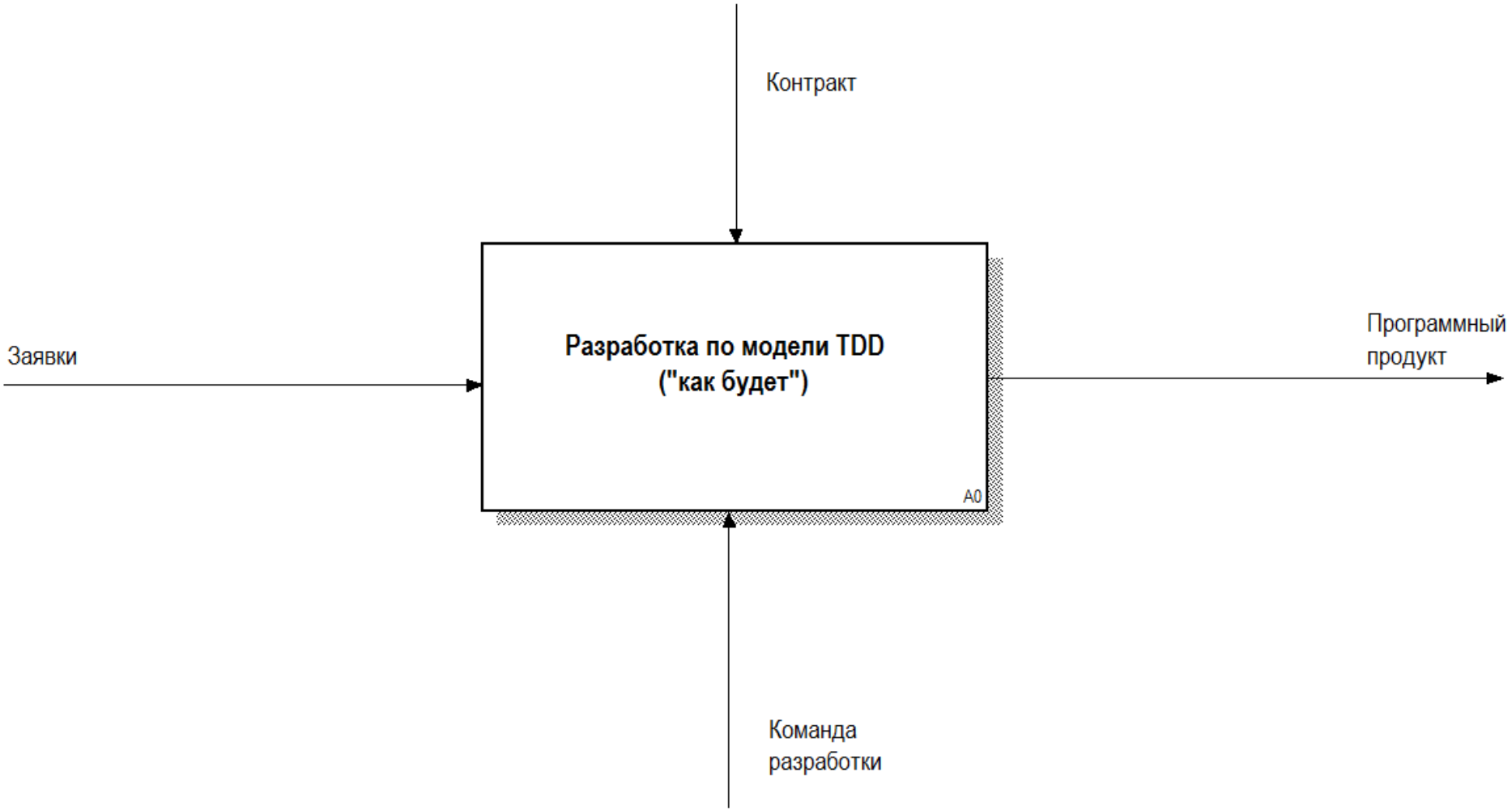
При выполнении преобразования необходимо дополнить (обогатить) модель данными обо всех потоках

В разработке, управляемой функциональностью или тестами, конфигурация программного и информационного обеспечения всегда может быть восстановлена с трудозатратами, сравнимыми с проектированием!

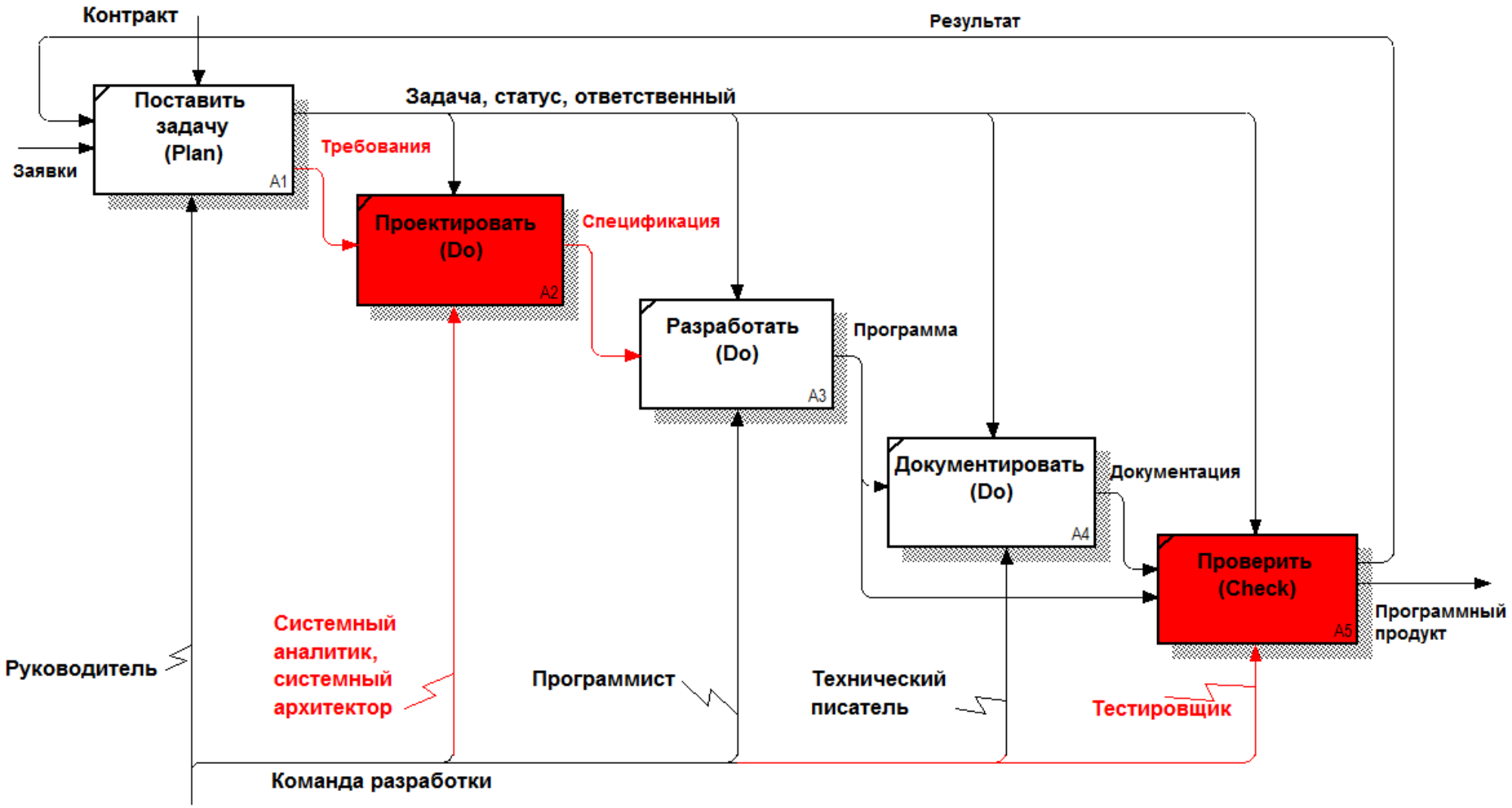
Реинжиниринг в FDD/TFD/TDD/BDD



Модель TDD


Контракт
Заявки
Разработка по модели TDD
("как будет")
A0
Программный
продукт
Команда
разработки

Модель TDD и команда



Рефакторинг в TDD/TFD

Разработка через тестирование (*test-driven development, **TDD***) — техника разработки программного обеспечения, которая основывается на повторении очень коротких циклов разработки: сначала пишется тест, покрывающий желаемое изменение, затем пишется код, который позволит пройти тест, и под конец проводится рефакторинг нового кода к соответствующим стандартам

TDD цикл включает в себя пять основных шагов:

1. Быстро добавить тест
2. Выполнить все тесты и увидеть, что новый тест "падает"
3. Выполнить небольшое изменение системы
4. Убедиться, что все тесты проходят
5. Выполнить **рефакторинг**, удаляя дублирование

В модели TDD **тест** всегда пишется **прежде** чем создается соответствующий **программный элемент**

Если далее не выполнять шаги 2, 4, 5 то получится модель **TFD** (разработка "вначале тест«, test first development).

Test-driven_development

Рефакторинг в TDD/TFD

Рефакторинг — процесс изменения **внутренней структуры** программы, не затрагивающий её **внешнего поведения** и имеющий целью:

- **облегчить понимание** её работы
- **устранить дублирование** кода
- облегчить **внесение изменений** в ближайшем будущем

Рефакторинг, рассматриваемый защитниками agile как техника "Или – Или", может быть лучше использован как "И" техника: он работает наилучшим образом, когда комбинируется с идеями, отвергаемыми аджилистами, в данном случае с предваряющим анализом

Никакое количество **рефакторинга** неспособно скорректировать **дефектную архитектуру.** Первичная обязанность любого проектировщика – идентифицировать фундаментальные абстракции, составляющие скелет архитектуры

Сделаете это нужным образом, и вам останется еще много работы, которую необходимо выполнить. Но если ошибиться на этом этапе, в конечном счете вам придется (метафору выберите сами) ставить латку на латку, тушить огонь керосином, лепить пластырь на пластырь

Рефакторинг в Agile (Интуит)

Рефакторинг в TDD/TFD

Код с запашко́м (**код с душко́м**, **дурно пахнущий код** англ. *code smell*) — термин, обозначающий код с признаками (запахами) проблем в системе

Был введён Кентом Беком и использован Мартином Фаулером в его книге *Рефакторинг. Улучшение существующего кода*

Запахи кода — это ключевые **признаки** необходимости рефакторинга

Существуют запахи, специфичные как для парадигм программирования, так и для конкретных языков

Основной проблемой, с которой сталкиваются разработчики при борьбе с запахами кода, является то, что критерии своевременности рефакторинга невозможно чётко формализовать без апелляции к эстетике и условному чувству прекрасного

Запахи кода — это **не набор** чётких **правил**, а описание **мест**, на которые нужно **обращать внимание** при рефакторинге. Они легко обнаруживаются, но при этом не во всех случаях свидетельствуют о проблемах

Рефакторинг в TDD/TFD

Комментарии

Часто комментарии играют роль «**дезодоранта**» кода, который появляется в нём лишь потому, что код плохой

Почувствовав потребность написать комментарий, попробуйте изменить структуру кода так, чтобы любые **комментарии стали излишними:**

- если для объяснения действий блока всё же требуется комментарий, попробуйте применить «**Выделение метода**» (Extract Method)
- если метод уже выделен, но по-прежнему нужен комментарий для объяснения его действия, воспользуйтесь «**Переименованием метода**» (Rename Method)
- если требуется изложить некоторые правила, касающиеся необходимого состояния системы, примените «**Введение утверждения**» (Introduce Assertion)

Код с запашком (Интуит)

Документирование программного кода

```
/*!
        \brief Дочерний класс
        \author Norserium
        \version 1.0
        \date Март 2015 года
        \warning Данный класс создан только в учебных целях

        Обычный дочерний класс, который отнас
*/
class Son : public Parent {
public:
        Son();
        ~Son();
};
```

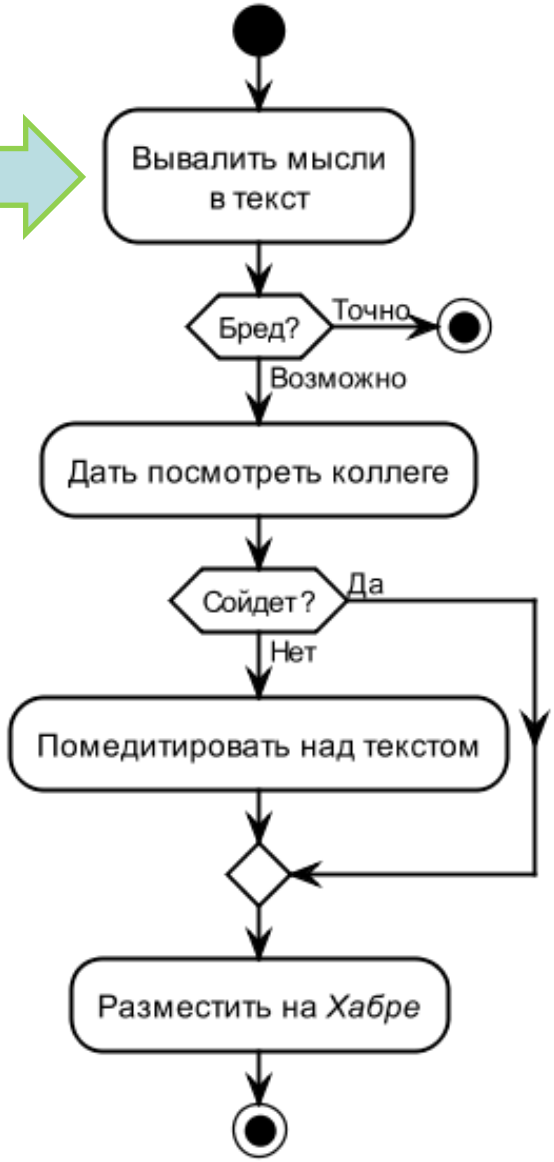
```
/*
    The main Math class
    Contains all methods for performing basic math functions
*/
/// <summary>
/// The main Math class.
/// Contains all methods for performing basic math functions.
/// </summary>
public class Math
{
    // Adds two integers and returns the result
    /// <summary>
    /// Adds two integers and returns the result.
    /// </summary>
    public static int Add(int a, int b)
    {
        // If any parameter is equal to the max value of an integer
        // and the other is greater than zero
        if ((a == int.MaxValue && b > 0) || (b == int.MaxValue && a > 0))
            throw new System.OverflowException();

        return a + b;
    }
}
```

Документирование в комментариях (Пример)

Документирование программного кода

```
start
:Вывалить мысли
в текст;
if (Бред?) then (Возможно)
    :Дать посмотреть коллеге;
    if (Сойдет?) then (Да)
    else (Нет)
        :Помедитировать над текстом;
    endif
    :Разместить на //Хабре//;
else (Точно)
    stop
endif
stop
```

PlantUML




```
if (a==true) { // если а верно
  b = b + 1;   // увеличить b
}
```

```
if (a) { // если найден элемент
  b++;   // указатель вперед на 1 слово
}
```

```
if (item_found) {
  next_item_pointer++;
}
```

PlantUML (Пример)