1. Назовите этапы разработки ПО

Этапы разработки программного обеспечения (ПО) могут варьироваться в зависимости от выбранной методологии, но в рамках курса «Роботизация бизнес-процессов (RPA) на платформе Атом.РИТА» и с учётом материалов лекций (в частности, Лекция 1, Лекция 6 и ссылки на BABOK и ГОСТ 34), можно выделить следующие основные этапы жизненного цикла ПО:

Согласно BABOK / SDLC (Software Development Life Cycle):

Планирование и анализ требований

— Выявление потребностей заказчика, сбор и анализ требований, определение целей проекта.

Проектирование (дизайн)

— Разработка архитектуры системы, создание технического задания, моделирование процессов (например, в нотациях BPMN, IDEF0).

Разработка (реализация)

— Написание кода, создание программного робота или другого ПО.

Тестирование

— Проверка корректности работы, составление сценариев и протоколов тестирования.

Внедрение (развертывание)

— Перенос решения в продуктивную среду, обучение пользователей, подготовка инструкций.

Сопровождение и поддержка

— Обеспечение стабильной эксплуатации, устранение ошибок, внесение изменений.

Вывод из эксплуатации

— Завершение использования системы или её замена.

Согласно ГОСТ 34.601–90 (для АСУ):

Формирование требований к АС

Разработка концепции АС
Техническое задание (ТЗ)
Эскизный проект
Технический проект
Рабочая документация
Ввод в действие
Сопровождение АС

2. На каких этапах разработки ПО задействован бизнес-аналитик?

Бизнес-аналитик задействован на всех ключевых этапах жизненного цикла разработки программного обеспечения (ПО), особенно на ранних и средних стадиях, где формируются требования и проектируется решение. Согласно материалам лекций по дисциплине «Роботизация бизнес-процессов (RPA) на платформе Атом.РИТА», его участие можно описать следующим образом:

1. Планирование и предпроектное обследование
Выявление бизнес-потребностей заказчика.
Определение целей проекта, границ и заинтересованных лиц.
Проведение интервью, SWOT-анализа, сбор первичной информации.
Формирование Offer-card — инициации идеи роботизации.
Источник: Лекции 1, 2, 5

2. Анализ требований
Документирование функциональных и нефункциональных требований.
Разработка User Story, Use Cases, описание в терминах BABOK.
Приоритизация требований (MoSCoW, RICE).
Управление ожиданиями заказчика и трассировка требований.
Источник: Лекции 2, 3, 4, 5

3. Проектирование (моделирование процессов)
Создание моделей «as-is» (как есть) и «to-be» (как должно быть).

Использование нотаций: BPMN, IDEF0, eEPC, UML Activity Diagram.
Подготовка Описания программного робота (ОПР) — ключевой документ в RPA.
Согласование ОПР с владельцем процесса и владельцем ИТ-ресурса.
Источник: Лекции 6, 7, 8, 11

4. Тестирование
Участие в подготовке сценариев тестирования и протоколов тестирования.
Совместное тестирование с разработчиком и заказчиком.
Проверка соответствия реализованного решения описанному в ОПР.
Источник: Лекция 11

5. Внедрение
Подготовка инструкции пользователя.
Обеспечение передачи знаний заказчику.
Участие в получении листов исполнения (ЛИ) на доступ к продуктивным системам.
Источник: Лекция 11

6. Сопровождение (частично)
Хотя основная роль бизнес-аналитика завершается после внедрения, на практике он может участвовать в:
Уточнении требований при доработках.
Анализе инцидентов, связанных с несоответствием бизнес-логики.
Поддержке владельца робота при изменении внешних условий (например, изменение интерфейса системы).
Источник: Лекция 2 (пересечение ролей аналитика и техподдержки)

Вывод
Бизнес-аналитик — ключевая фигура на этапах от инициации до внедрения, особенно в RPA-проектах, где точность описания процесса напрямую влияет на успешность автоматизации. Его задача — гарантировать, что разрабатываемое решение соответствует реальным бизнес-нуждам и может быть корректно реализовано программным роботом.

3. В чем ключевые различия между водопадной и гибкими методологиями?

Ключевые различия между водопадной (Waterfall) и гибкими (Agile) методологиями разработки ПО заключаются в подходе к управлению проектом, последовательности выполнения работ, гибкости изменений и взаимодействию с заказчиком. Ниже приведено сравнение по основным аспектам:

1. Последовательность этапов

Водопадная модель:

— Этапы строго последовательны: анализ → проектирование → реализация → тестирование → внедрение → сопровождение.

— Переход к следующему этапу возможен только после полного завершения предыдущего.

— Обратная связь или возврат на предыдущий этап затруднены и дорогостоящи.

Гибкие методологии (Agile, Scrum, Kanban и др.):

— Проект делится на итерации (спринты), каждая из которых включает все этапы ЖЦ ПО (анализ, разработка, тестирование).

— Работа выполняется инкрементально и итеративно, с постоянной обратной связью.

2. Изменение требований

Waterfall:

— Требования фиксируются на начальном этапе и почти не подлежат изменению.

— Любые изменения в ходе проекта считаются отклонениями и требуют формального пересмотра документации.
Agile:
— Изменения приветствуются даже на поздних стадиях разработки.
— Гибкость — одно из ключевых ценностей Agile (см. «Agile Manifesto»).
3. Взаимодействие с заказчиком
Waterfall:
— Заказчик участвует в основном на этапе сбора требований и при приемке конечного продукта.
— Минимальное взаимодействие в процессе разработки.
Agile:
— Заказчик (или Product Owner) вовлечен постоянно: участвует в планировании спринтов, ревью, приоритизации задач.
— Регулярная демонстрация работающего функционала.
4. Документирование
Waterfall:
— Большой упор на подробную документацию (ТЗ, спецификации, протоколы и т.д.).
— Документация — основной артефакт передачи знаний.
Agile:
— Документация минимизирована в пользу работающего ПО.
— Акцент на коммуникацию и живые артефакты (например, User Stories, бэклог).

5. Риски и обратная связь
Waterfall:
— Риск обнаружения критических ошибок — ближе к концу проекта.
— Обратная связь поставляется редко и поздно.
Agile:
— Риски выявляются рано и часто благодаря регулярным итерациям.
— Возможность корректировки курса на каждом этапе.
6. Применимость
Waterfall эффективна при:
Четко определённых, стабильных требованиях.
Проектах с жёсткими регуляторными или нормативными ограничениями (например, ГОСТ 34 в госсекторе).
Низкой вероятности изменений (например, встроенные системы, оборонка).
Agile предпочтителен при:
Неопределённых или быстро меняющихся требованиях.
Быстром выводе MVP (минимально жизнеспособного продукта).
Высокой вовлечённости заказчика.
Связь с курсом «Роботизация бизнес-процессов (RPA)»
В материалах лекций (Лекция 1, Лекция 2) упоминаются:

BABOK и SDLC, где водопадная модель представлена как классический подход.
User Story, MoSCoW, итеративная приоритизация — типичные практики Agile.
В RPA-проектах часто используется гибридный подход: аналитическая часть (ОПР) может быть более

структурированной (похожей на Waterfall), но сама разработка и тестирование — итеративны (Agile).

Критерий	Waterfall	Agile
Последовательность	Линейная, строгая	Итеративная, цикличная
Изменение требований	Затруднено	Приветствуется
Роль заказчика	Минимальна	Активна
Документация	Подробная	Минимальная
Обратная связь	В конце	На каждой итерации
Управление рисками	Позднее выявление	Раннее и частое

Выбор методологии зависит от контекста проекта, стабильности требований и организационной культуры.

4. Какие гибкие методологии знаете?
Среди гибких методологий разработки ПО (Agile), наиболее известные и применяемые в практике, включая контекст курса «Роботизация бизнес-процессов (RPA) на платформе Атом.РИТА», следующие:

1. Scrum
Итеративная фреймворк-методология.
Работа организуется в спринты (обычно 2–4 недели).
Есть роли: Product Owner, Scrum Master, Команда разработки.
Артефакты: Product Backlog, Sprint Backlog, Increment.
Мероприятия: планирование спринта, ежедневные стендапы, ревью, ретроспектива.
Упоминается в лекциях косвенно через практики приоритизации (MoSCoW, User Story), характерные для Scrum.

2. Kanban
Визуальная система управления потоком работ.
Использует доску с колонками («To Do», «In Progress», «Done»).
Нет фиксированных итераций — работа непрерывна.
Ограничение Work In Progress (WIP) для повышения эффективности.
Ссылка на Kanban присутствует в Лекции 2: «https://www.atlassian.com/ru/agile/kanban»

3. Extreme Programming (XP) — Экстремальное программирование
Фокус на техническое качество кода и частые релизы.
Практики: парное программирование, тестирование до кода (TDD), непрерывная интеграция, рефакторинг.
Подходит для проектов с быстро меняющимися требованиями.
4. Lean Software Development (Бережливая разработка)
Произошла от Lean-производства (Toyota).
Принципы: устранение потерь, усиление обучения, быстрая доставка, уважение к людям.
Акцент на ценность для клиента и минимизацию ненужной работы.
5. Feature-Driven Development (FDD)
Ориентирована на функциональные возможности (features).
Включает этапы: моделирование предметной области, планирование по функциям, проектирование и реализация по функциям.
6. Dynamic Systems Development Method (DSDM)
Одна из первых Agile-методологий (1990-е).

Жёсткие временные рамки, фиксированный бюджет, гибкие требования.
Активное участие заказчика.
В контексте RPA и курса:
Хотя в лекциях прямо не перечисляются все гибкие методологии, практики Agile активно используются:

User Story и Use Cases (Лекции 4, 5)
Приоритизация по MoSCoW (Лекция 5)
Итеративная разработка и тестирование программных роботов (Лекция 11)
Вовлечение заказчика на всех этапах
Это указывает на применение гибридных или адаптированных Agile-подходов, особенно Scrum и Kanban, в проектах RPA.

Вывод:
Основные гибкие методологии:

Scrum
Kanban
XP
Lean
FDD
DSDM
В RPA-проектах чаще всего применяются элементы Scrum и Kanban, адаптированные под специфику автоматизации бизнес-процессов.

5. Какие основные задачи выполняет бизнес-аналитик на проекте?

Основные задачи бизнес-аналитика на проекте, согласно материалам курса «Роботизация бизнес-процессов (RPA) на платформе Атом.РИТА», можно сгруппировать по ключевым направлениям деятельности:

1. Выявление и анализ бизнес-потребностей
Проведение интервью, опросов, SWOT-анализа.
Определение целей проекта и ожидаемых результатов.
Формулирование бизнес-требований (why — зачем это нужно бизнесу?).
Источник: Лекции 1, 2, 3

2. Сбор, документирование и управление требованиями
Разработка функциональных и нефункциональных требований.
Использование техник описания:
User Story (по INVEST-критериям),
Use Cases (сценарии взаимодействия пользователя и системы).
Приоритизация требований (например, по методу MoSCoW или RICE).
Трассировка требований — связь между источником, реализацией и тестированием.
Источник: Лекции 3, 4, 5

3. Моделирование бизнес-процессов
Создание моделей «as-is» (как есть) и «to-be» (как должно быть).
Применение нотаций:
BPMN,
IDEF0,
eEPC,
UML Activity Diagram.
Визуализация процессов для согласования с заказчиком и разработчиками.
Источник: Лекции 6, 7, 8

4. Подготовка документов для RPA-проектов
Формирование Offer-card — инициация идеи роботизации.
Разработка Описания программного робота (ОПР) — ключевой документ в RPA:
Назначение робота,
Пошаговый алгоритм работы,
Взаимодействие с ИТ-ресурсами,
Бизнес-ограничения.
Подготовка сценариев и протоколов тестирования, инструкций пользователя.
Источник: Лекции 9, 11

5. Расчёт эффекта от роботизации
Оценка трудозатрат до и после внедрения робота.
Расчёт экономического эффекта в ПРМ (полных рабочих месяцах) и рублях.
Обоснование целесообразности автоматизации.
Источник: Лекция 11

6. Координация и коммуникация
Взаимодействие с заинтересованными сторонами: владельцем процесса, владельцем ИТ-ресурса, разработчиками, службой поддержки.
Управление ожиданиями заказчика.
Обеспечение согласования решений между бизнесом и ИТ.
Источник: Лекции 2, 11

7. Участие в тестировании и внедрении
Совместное тестирование с разработчиком и заказчиком.
Подготовка ЛИ (листов исполнения) на доступ к тестовым и продуктивным системам.
Передача решения в эксплуатацию.
Источник: Лекция 11

Итог

Бизнес-аналитик выступает как мост между бизнесом и технологиями. Его главная цель — точно понять, что нужно заказчику, и передать это разработчикам в понятной, структурированной форме, особенно в контексте RPA, где малейшая неточность в описании процесса может привести к сбою робота.

Как сказано в лекции 2:
«Аналитик — связующее звено между Клиентом и Разработчиком».

6. Какими знаниями и навыками должен обладать бизнес

Бизнес-аналитик должен обладать сочетанием технических (hard skills) и межличностных (soft skills) компетенций, позволяющих эффективно выявлять, анализировать и документировать бизнес-требования, а также выступать связующим звеном между заказчиком и разработчиками. Ниже приведены ключевые знания и навыки на основе материалов курса «Роботизация бизнес-процессов (RPA) на платформе Атом.РИТА».

- Hard Skills (технические навыки)

Знание жизненного цикла ПО (SDLC)

— Понимание этапов разработки: от анализа до сопровождения.
— Знание методологий: Waterfall, Agile (Scrum, Kanban).
Моделирование бизнес-процессов
— Владение нотациями: BPMN, IDEF0, eEPC, UML Activity Diagram.
— Умение строить модели «as-is» (как есть) и «to-be» (как должно быть).
Сбор и документирование требований
— Формулирование бизнес-, функциональных и нефункциональных требований.
— Использование форматов: User Story (по INVEST), Use Cases.
— Работа с ГОСТ 34 (в госсекторе и крупных корпорациях).
Приоритизация и трассировка требований
— Применение методов: MoSCoW, RICE, Feature Bucket.
— Обеспечение связи между источником требования, его реализацией и тестированием.
Основы RPA и автоматизации

— Понимание, что такое программный робот и какие процессы подлежат роботизации.
— Умение составлять Описание программного робота (ОПР) и Offer-card.
Работа с инструментами
— Системы управления проектами (Jira, YouGile и др.).
— Инструменты моделирования: Draw.io, Bizagi, ARIS, Camunda и др.
Базовое понимание ИТ и интеграций
— Знание принципов работы с API, базами данных, офисными приложениями (Excel, Outlook).
— Понимание ограничений UI-автоматизации и возможностей OCR.

- Soft Skills (личностные и коммуникативные навыки)

Коммуникация и ведение переговоров
— Умение слушать заказчика, задавать правильные вопросы, уточнять неоднозначности.
— Ведение интервью по методике SMART.
Управление ожиданиями
— Настройка реалистичных ожиданий по срокам, функционалу и эффекту.

— Работа с матрицей стейкхолдеров.
Аналитическое и системное мышление
— Декомпозиция сложных процессов на простые шаги.
— Выявление корневых причин проблем, а не только симптомов.
Управление временем и приоритетами
— Использование матрицы Эйзенхауэра.
— Способность фокусироваться на задачах с высокой ценностью.
Командная работа и конфликтология
— Эффективное взаимодействие с разработчиками, тестировщиками, владельцами процессов.
— Разрешение разногласий между бизнесом и ИТ.
Стрессоустойчивость и адаптивность
— Работа в условиях неопределенности и изменяющихся требований.

- Дополнительно (в контексте RPA)

Умение рассчитывать эффект от роботизации в ПРМ (полных рабочих месяцах) и рублях.

Понимание ролей в RPA-проекте: владелец процесса, владелец ИТ-ресурса, служба поддержки ПР и др.
Навык подготовки сопутствующей документации: сценарии тестирования, инструкции пользователя, листы исполнения (ЛИ).
💡 Цитата из лекций:
«Аналитик — связующее звено между Клиентом и Разработчиком» (Лекция 2).

Вывод
Бизнес-аналитик — это гибрид бизнес-эксперта и технического специалиста, который должен:

глубоко понимать предметную область,
говорить на языке и бизнеса, и ИТ,
уметь структурировать хаос требований в четкий, реализуемый план.
Эти компетенции особенно важны в проектах RPA, где даже малейшая неточность в описании процесса может привести к сбою робота.

7. Кто такой ЛПР? Какие еще категории заинтересованных лиц можете выделить?

ЛПР — это лицо, принимающее решение (от англ. Decision Maker).
В контексте проектов по роботизации (RPA) и бизнес-анализа ЛПР — это человек или группа лиц, обладающие полномочиями утверждать:

инициацию проекта,
выделение бюджета,
выбор решения,
запуск или остановку разработки/внедрения программного робота.
Как правило, ЛПР — это руководитель подразделения, владелец процесса, владелец бюджета или топ-менеджер, заинтересованный в результате автоматизации.

Другие категории заинтересованных лиц (стейкхолдеров)
Согласно материалам лекций (Лекция 2, Лекция 4, Лекция 11), можно выделить следующие категории:

1. Владелец процесса
— Отвечает за качество и своевременность выполнения бизнес-процесса.
— Определяет, что должно быть автоматизировано и какой результат ожидается.

2. Владелец ИТ-ресурса
— Отвечает за использование и развитие информационной системы, с которой взаимодействует робот (например, 1С, SAP, СЭД).
— Даёт согласие на интеграцию робота с системой.

3. Функциональный заказчик / Инициатор (Заказчик ПР)

— Тот, кто формулирует идею роботизации и подаёт Offer-card.
— Может совпадать с владельцем процесса, но не всегда.

4. Конечный пользователь
— Сотрудник, для которого создаётся робот или чья работа изменяется в результате автоматизации.
— Может как выигрывать (освобождение от рутины), так и терять часть функций.

5. Разработчик ПР (СРПР — Служба разработки ПР)
— Реализует робота на платформе Атом.РИТА на основе ОПР.

6. Служба поддержки ПР (СППР)
— Обеспечивает техническую поддержку, мониторинг и устранение инцидентов в продуктивной среде.

7. Администратор ИБ ИТ-ресурса

— Отвечает за информационную безопасность.
— Проверяет, не нарушает ли робот политики безопасности (например, запросы на административные права).

8. Эксперт функционального направления (ЭФН)
— Обеспечивает соответствие алгоритма робота существующим бизнес-правилам в ИС.
— Предотвращает конфликты с уже автоматизированными процессами.

9. Менеджер ПР
— Координирует работы по созданию/доработке робота, контролирует сроки и ресурсы.

10. Владелец учётной записи программного робота
— Формальный «владелец» цифрового сотрудника; обычно сотрудник СППР или СРПР.

Визуализация: Матрица стейкхолдеров (Лекция 5)
Стейкхолдеры также могут классифицироваться по:

Уровню влияния (высокий / низкий),
Уровню интереса (высокий / низкий).
Это помогает определить стратегию взаимодействия:

Управлять внимательно (высокое влияние + высокий интерес) → ЛПР, владелец процесса.
Держать в курсе (низкое влияние + высокий интерес) → конечные пользователи.
Информировать минимально (низкое влияние + низкий интерес) → некоторые ИТ-администраторы.
Вывод
ЛПР — ключевой стейкхолдер, без одобрения которого проект не стартует. Однако успешная роботизация требует вовлечения всех категорий

заинтересованных лиц, каждая из которых играет свою роль на разных этапах жизненного цикла ПР.

8. Перечислите методы сбора требований

Методы сбора требований — это техники, используемые бизнес-аналитиком для выявления, уточнения и документирования потребностей заинтересованных лиц. Согласно материалам курса «Роботизация бизнес-процессов (RPA) на платформе Атом.РИТА» (в частности, Лекции 2, 3, 4, 5), можно выделить следующие основные методы:

1. Интервью
Индивидуальная беседа с заинтересованным лицом (например, владельцем процесса).
Позволяет глубоко понять контекст, выявить скрытые потребности.
Рекомендуется проводить по технике SMART (Лекция 5): цели интервью должны быть конкретными, измеримыми, достижимыми, релевантными и ограниченными по времени.
2. Опросы и анкетирование
Массовый сбор информации от группы пользователей.
Эффективны при работе с большим количеством стейкхолдеров.
Могут быть как открытыми (свободные ответы), так и закрытыми (выбор из вариантов).
3. Наблюдение (Work shadowing)
Аналитик наблюдает за выполнением сотрудником его рабочих задач в реальном времени.
Особенно полезен при автоматизации рутинных операций в RPA.
Позволяет выявить неформальные действия, которые не отражены в регламентах.
4. Анализ документации
Изучение существующих регламентов, инструкций, форм, отчётов, ТЗ, политик компании.
Помогает понять текущее состояние процесса (модель «as-is»).
5. Мозговой штурм (Brainstorming)

Групповая генерация идей в присутствии заказчика, разработчиков и других стейкхолдеров.
Используется на ранних этапах для выявления возможных решений.
6. Workshops (рабочие сессии)
Организованные встречи с участием ключевых заинтересованных лиц.
Цель — совместно описать процесс, согласовать требования, определить приоритеты.
Часто используются при создании моделей в нотациях BPMN, IDEF0, eEPC.
7. Прототипирование
Создание упрощённой версии будущего решения (например, макет формы или экрана).
Позволяет быстро получить обратную связь от заказчика.
8. Анализ аналогов (Benchmarking)
Изучение решений в других подразделениях или компаниях.
Помогает избежать ошибок и использовать лучшие практики.
9. Использование User Story и Use Cases
Хотя это скорее форматы документирования, они также служат методом структурированного сбора требований:
User Story: «Как [роль], я хочу [цель], чтобы [выгода]».
Use Case: пошаговое описание взаимодействия пользователя с системой.
Источник: Лекции 4, 5

10. SWOT-анализ
Косвенный метод, помогающий выявить сильные и слабые стороны текущего процесса, а также возможности и угрозы, связанные с его автоматизацией.
Источник: Лекция 1

Вывод
Выбор метода зависит от:

уровня вовлечённости стейкхолдеров,
степени формализации процесса,
наличия документации,
сложности предметной области.
В проектах RPA чаще всего комбинируют интервью, наблюдение и анализ документации, так как требуется точное понимание шагов, выполняемых человеком в UI-системах.

9. Как подготовиться к выявлению требований?

Подготовка к выявлению требований — важнейший этап работы бизнес-аналитика, от которого зависит качество дальнейшего анализа и успешность проекта. Согласно материалам курса «Роботизация бизнес-процессов (RPA) на платформе Атом.РИТА» (Лекции 2, 3, 4, 5), подготовка включает следующие ключевые шаги:

1. Определить цели и границы проекта
Чётко сформулировать бизнес-потребность: зачем нужен проект, какую проблему он решает.
Установить границы проекта (project scope): что входит в проект, а что — нет.
Определить критерии успеха (например, снижение трудозатрат, устранение ошибок).
Источник: Лекция 3, Лекция 4

2. Идентифицировать заинтересованных лиц (стейкхолдеров)
Составить список всех участников: владелец процесса, владелец ИТ-ресурса, конечные пользователи, ЛПР и др.
Построить матрицу стейкхолдеров по уровням влияния и интереса (Лекция 5).
Определить, кто будет основным источником требований.
Источник: Лекция 2, Лекция 5

3. Изучить контекст и существующую документацию
Проанализировать:
регламенты,
инструкции,
формы,
отчёты,
текущие ИТ-системы.
Это помогает понять модель «as-is» и избежать дублирования или противоречий.
Источник: Лекция 6, Лекция 7

4. Выбрать методы сбора требований

На основе контекста выбрать подходящие техники:

Интервью (по SMART-целям),
Наблюдение (work shadowing) — особенно важно в RPA,
Опросы/анкетирование,
Workshop’ы для совместного моделирования,
Анализ документов.
Источник: Лекция 2, Лекция 5

5. Подготовить инструменты и материалы
Шаблоны для:
User Story,
Use Cases,
протоколов интервью,
матрицы приоритизации (MoSCoW, RICE).
Инструменты моделирования (например, Draw.io, Bizagi) — для быстрой визуализации процессов.
Источник: Лекция 4, Лекция 8

6. Сформулировать цели встреч/интервью (по SMART)
Цель должна быть:

S — конкретной (Specific),
M — измеримой (Measurable),
A — достижимой (Achievable),
R — релевантной (Relevant),
T — ограниченной по времени (Time-bound).
Пример:

«За 60 минут интервью с владельцем процесса выявить все шаги обработки входящих счетов и определить точки интеграции с СЭД».

Источник: Лекция 5

7. Продумать сценарии взаимодействия
Подготовить вопросы заранее.

Учесть возможные сопротивления со стороны сотрудников (страх потери работы, недоверие к автоматизации).
Продумать, как мотивировать участников делиться информацией.
Источник: Лекция 2

8. Уточнить технические ограничения
Доступны ли системы для робота?
Есть ли API или только UI?
Требуется ли OCR?
Какие есть ограничения по безопасности?
Источник: Лекция 11

Вывод
Подготовка к выявлению требований — это не просто «прийти и спросить», а структурированный процесс, включающий:

анализ контекста,
планирование коммуникаций,
выбор методов,
подготовку инструментов.
Хорошая подготовка позволяет:

сократить количество итераций,
избежать упущенных требований,
повысить доверие заказчика,
заложить основу для корректного Описания программного робота (ОПР).
Как говорится в лекциях:
«Без ТЗ — результат хз...» (Лекция 3).

10. Каковы ваши действия после выявления требований?

После выявления требований бизнес-аналитик выполняет ряд последовательных действий, направленных на структурирование, документирование, согласование и передачу требований для дальнейшей реализации. Согласно материалам курса «Роботизация бизнес-процессов (RPA) на платформе Атом.РИТА» (Лекции 2–6, 11), эти действия включают:

1. Анализ и структурирование собранных данных
Устранение противоречий и дублирования.
Группировка требований по категориям:
— бизнес-требования (зачем?),
— функциональные (что делает система?),
— нефункциональные (SLA, производительность, безопасность).
Декомпозиция сложных процессов на подпроцессы.
Источник: Лекция 3

2. Моделирование бизнес-процессов
Построение модели «as-is» (как есть сейчас).
Разработка целевой модели «to-be» (как должно быть после автоматизации).
Использование нотаций: BPMN, IDEF0, eEPC, UML Activity Diagram.
Источник: Лекции 6, 7, 8

3. Документирование требований
Выбор подходящего формата в зависимости от контекста:

User Story (по INVEST-критериям) — для гибких проектов.
Use Cases — для детального описания сценариев взаимодействия.
Техническое задание (ТЗ) — в соответствии с ГОСТ 34 (в госсекторе или крупных корпорациях).
Описание программного робота (ОПР) — специфический документ для RPA-проектов.
Источник: Лекции 4, 5, 11

4. Приоритизация требований
Применение методов:

MoSCoW: Must, Should, Could, Would.
RICE: Reach, Impact, Confidence, Effort.
Feature Bucket: Metric Movers, Customer Requests, Customer Delight.
Источник: Лекция 5

5. Согласование с заинтересованными лицами

Проведение workshop’ов или встреч для утверждения:
модели «to-be»,
ОПР,
User Story / Use Cases.
Получение формального одобрения от владельца процесса, владельца ИТ-ресурса и ЛПР.
Источник: Лекции 2, 11

6. Подготовка к передаче разработчикам
Формирование полного пакета аналитической документации:
ОПР,
сценарии тестирования,
протоколы,
листы исполнения (ЛИ) на доступ к тестовым системам.
Участие в планировании разработки (например, в рамках Scrum или Kanban).
Источник: Лекция 11

7. Обеспечение трассировки требований
Установление связей между:
источником требования,
его реализацией в коде,
тест-кейсами.
Это позволяет отслеживать, все ли требования реализованы и протестированы.
Источник: Лекция 4

Вывод
Действия после выявления требований — это не просто «записать и отправить», а целый цикл аналитической работы, направленный на превращение неструктурированных пожеланий заказчика в четкий, реализуемый и проверяемый технический документ, пригодный для разработки программного робота.

Как говорится в лекции 3:

«Без ТЗ – результат хз...»

11. Уровни требований

Согласно материалам курса «Роботизация бизнес-процессов (RPA) на платформе Атом.РИТА» (в частности, Лекция 3), требования в проекте разделяются на уровни, отражающие их происхождение, назначение и степень детализации. Выделяют следующие основные уровни требований:

1. Бизнес-требования (Business Requirements)
Отвечают на вопрос: «Зачем? Почему мы это делаем? Какой бизнес-цель хотим достичь?»
Формулируются заказчиком или владельцем процесса.
Описывают стратегические цели, ценность для бизнеса, ожидаемый эффект.
Пример:
«Сократить время обработки входящих счетов с 2 часов до 15 минут в день».

Источник: Лекция 3, слайд 11

2. Функциональные требования (Functional Requirements)
Отвечают на вопрос: «Что система должна делать?»
Описывают поведение системы, действия, которые она должна выполнять.
Часто формулируются через User Story или Use Cases.
Пример:
«Система должна автоматически извлекать данные из PDF-счета и вносить их в ERP».

Источник: Лекции 3, 4

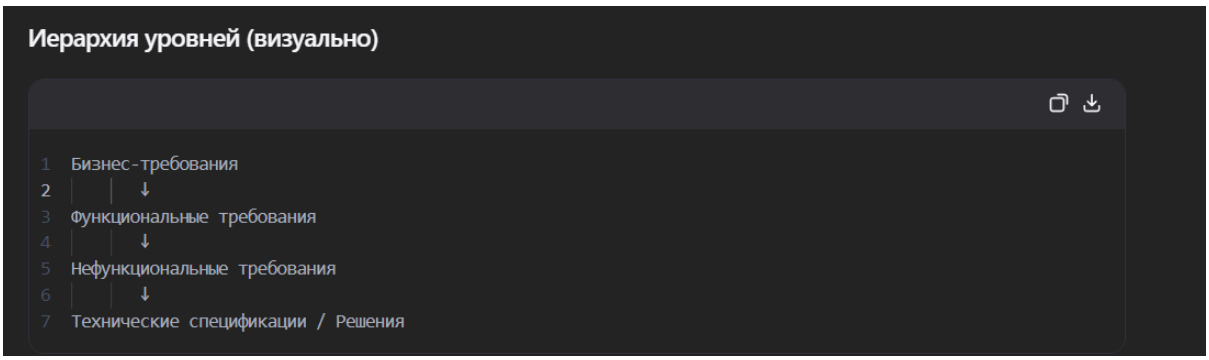
3. Нефункциональные требования (Non-functional Requirements / SLA)
Отвечают на вопрос: «Как система должна это делать?»

Определяют качественные характеристики: производительность, надёжность, безопасность, удобство использования и т.д.
Часто выражаются в виде ограничений или метрик.
Примеры:
«Робот должен обрабатывать 100 документов за 10 минут»,
«Доступ к системе — только по двухфакторной аутентификации»,
«Время простоя робота не более 1% в месяц».
Источник: Лекция 3, слайд 11

Дополнительно: Уровень решений (Solution Level)
Хотя это не уровень требований как таковой, в лекциях подчёркивается важность различать:

Требование — что нужно,
Решение — как это будет реализовано.
Пример из Лекции 3:

Потребность: «Морякам нужна еда на 30 дней».
Требование: «Продукты должны сохраняться при +30°C и высокой влажности».
Решение: «Использовать консервы и вакуумную упаковку».
Источник: Лекция 3, слайд 7



Вывод
Правильное разделение требований по уровням позволяет:

избежать путаницы между «зачем» и «как»,

обеспечить полноту и согласованность аналитики,
корректно передать задачу разработчикам.
Как говорится в лекции:

«Без ТЗ – результат хз...» (Лекция 3, слайд 3).

12. Характеристик требований

Характеристики (или качественные признаки) требований определяют, насколько они пригодны для использования в проекте. Согласно материалам курса «Роботизация бизнес-процессов (RPA) на платформе Атом.РИТА» (в частности, Лекция 3 и связанные темы), а также общепринятым практикам (например, BABOK, INVEST), хорошие требования должны обладать следующими характеристиками:

✅ 1. Понятность (Clarity)
Требование должно быть сформулировано однозначно и ясно, без двусмысленности.

Пример плохого: «Система должна быть удобной».
Пример хорошего: «Пользователь должен иметь возможность войти в систему за ≤ 2 клика».

✅ 2. Измеримость (Measurability)
Должна быть возможность проверить выполнение требования объективно.

Например: «Время загрузки страницы — не более 2 секунд».

✅ 3. Тестируемость (Testability)
Требование должно быть таким, чтобы можно было создать тест-кейс для его проверки.

Если нельзя протестировать — значит, нельзя убедиться, что оно реализовано.

✅ 4. Достижимость (Feasibility)
Требование должно быть реализуемо в рамках имеющихся ресурсов, технологий и сроков.

Нереалистичное требование: «Робот должен читать мысли пользователя».

✅ 5. Актуальность / Ценность (Relevance / Value)
Требование должно приносить пользу бизнесу и соответствовать целям проекта.

В RPA это особенно важно: автоматизируем только то, что даёт эффект (экономию ПРМ).

✅ 6. Непротиворечивость (Consistency)
Требования не должны противоречить друг другу или существующим правилам/системам.

Например, нельзя одновременно требовать «полный доступ к данным» и «строгую защиту персональных данных без исключений» без уточнения контекста.

✅ 7. Полная независимость или контролируемая зависимость (Independence)
Идеально, если требования минимально зависят друг от друга (особенно в Agile). Это позволяет гибко менять приоритеты.

Соответствует принципу I в INVEST: Independent.

✅ 8. Атомарность (Atomicity)
Каждое требование должно описывать одну функцию или одно условие.

Не: «Система должна принимать заказ и отправлять уведомление».
А: два отдельных требования — на приём заказа и на отправку уведомления.

✅ 9. Следуемость / Трассируемость (Traceability)
Должна быть возможность проследить, откуда взялось требование (от какого стейкхолдера, документа, цели) и куда оно вошло (в код, тест, ОПР).

Особенно важно в RPA для согласования с владельцем процесса и ИТ-ресурса.

💡 Примечание из лекций:
В Лекции 3 подчеркивается разница между:

Потребностью («зачем?»),
Требованием («что нужно?»),
Решением («как сделать?»).
Хорошее требование — это не решение, а чёткое, измеримое описание того, что система должна делать или обеспечивать.

Вывод
Характеристики качественного требования можно запомнить по мнемонике SMART + T:

S — Specific (конкретное)
M — Measurable (измеримое)
A — Achievable (достижимое)
R — Relevant (релевантное)
T — Testable (тестируемое)
Эти принципы напрямую применяются при подготовке Описания программного робота (ОПР) и других аналитических документов в RPA-проектах.

13. К каким последствиям приводят «плохие» требования?

Плохие» требования — это неточные, неполные, противоречивые, неизмеримые или нереалистичные формулировки потребностей заказчика. Согласно материалам курса «Роботизация бизнес-процессов (RPA) на платформе Атом.РИТА» (в частности, Лекция 3, слайд 3: «Без ТЗ – результат хз...»), такие требования ведут к серьёзным негативным последствиям на всех этапах проекта.

🔴 Основные последствия «плохих» требований:
1. Неверная реализация функционала
Разработчик делает не то, что нужно бизнесу, а то, что понял из расплывчатого описания.
Пример: вместо автоматизации обработки PDF-счетов робот настроен на Excel — потому что в ОПР не уточнили формат.
2. Увеличение сроков и стоимости проекта

Необходимость переделок, уточнений, повторного тестирования.
Рост трудозатрат аналитика, разработчика и заказчика.
3. Конфликты между заказчиком и исполнителем
Заказчик недоволен: «Я так не просил!»
Исполнитель возражает: «Вы так написали!»
Нарушается доверие и эффективность взаимодействия.
4. Сбои в работе программного робота
В RPA особенно критично: если шаг процесса описан неточно (например, «нажать кнопку» без указания её расположения или условий появления), робот аварийно завершится.
Это ведёт к простою, необходимости ручного вмешательства, потере эффекта от автоматизации.
5. Невозможность тестирования
Если требование не тестируемо (например: «система должна быть удобной»), нельзя проверить, выполнено ли оно.
Это подрывает качество и надёжность решения.
6. Отсутствие измеримого эффекта
Без чётких метрик (время, объём, частота) невозможно рассчитать экономический эффект или ПРМ (полные рабочие месяцы).
Руководство не видит ценности проекта → снижается поддержка RPA-инициатив.
7. Риск отказа от внедрения
Если после запуска выясняется, что робот не решает реальную проблему, его отключают, а репутация RPA-направления страдает.
💡 Пример из лекций:
«Без ТЗ – результат хз...» (Лекция 3, слайд 3) — эта фраза подчеркивает, что отсутствие или низкое качество требований делает исход проекта непредсказуемым.

✅ Как избежать:
Формулировать требования по принципу SMART (конкретные, измеримые, достижимые, релевантные, ограниченные по времени).
Использовать User Story и Use Cases с чёткими критериями приёмки.
Согласовывать Описание программного робота (ОПР) с владельцем процесса и владельцем ИТ-ресурса.
Проводить совместное тестирование до внедрения.

Вывод:
«Плохие» требования — главная причина провала IT-проектов, особенно в RPA, где точность описания процесса напрямую определяет работоспособность робота. Качественный анализ и документирование требований — основа успеха.

14. Структура ТЗ по ГОСТ 34

Согласно материалам курса «Роботизация бизнес-процессов (RPA) на платформе Атом.РИТА» и стандарту ГОСТ 34.602–89 «Техническое задание на создание автоматизированной системы», структура Технического задания (ТЗ) включает следующие обязательные разделы:

1. Общие сведения
Наименование системы.
Основание для разработки (например, приказ, договор).
Наименование организаций: заказчика, разработчика, эксплуатирующей организации.
Ссылки на нормативные документы.
2. Назначение и цели создания (развития) системы
Описание задач, которые должна решать система.
Цели автоматизации (повышение эффективности, снижение ошибок и т.д.).
3. Характеристики объектов автоматизации
Описание предметной области.
Перечень подразделений, процессов, функций, подлежащих автоматизации.
Текущее состояние ИТ-инфраструктуры (если применимо).
4. Требования к системе
Этот раздел является основным и делится на подразделы:

4.1 Требования к функциональным характеристикам
— Перечень автоматизируемых функций, описание их логики.
4.2 Требования к надёжности
— Уровень отказоустойчивости, время восстановления и т.п.
4.3 Условия эксплуатации
— Требования к рабочему месту, ПО, оборудованию.
4.4 Требования к информационной и программной совместимости
— Интеграции с другими системами, форматы обмена данными.
4.5 Требования к алгоритмам (при необходимости)
4.6 Требования к техническим средствам

— Серверы, клиентские станции, периферия.
4.7 Требования к программному обеспечению
— ОС, СУБД, библиотеки и т.д.
4.8 Требования к математическому обеспечению
4.9 Требования к информационному обеспечению
— Структура БД, справочники, классификаторы.
4.10 Требования к организационному обеспечению
4.11 Требования к составу и параметрам технико-экономических показателей
5. Состав и содержание работ по созданию системы
Этапы разработки (в соответствии с ГОСТ 34.601).
Перечень документов, подлежащих разработке.
Порядок сдачи-приёмки работ.
6. Порядок контроля и приёмки системы
Виды испытаний (предварительные, приёмочные).
Состав комиссии.
Форма актов и протоколов.
7. Требования к составу и содержанию работ по подготовке объекта автоматизации к вводу системы в действие
Подготовка помещения, персонала, данных.
Миграция информации из старых систем.
8. Требования к документированию
Перечень подлежащей разработке документации:
Руководство пользователя,
Руководство администратора,
Программа и методика испытаний и др.
9. Источники разработки
Используемые материалы, аналоги, существующие решения.
Источник: Лекция 6 (слайды 5–6), ГОСТ 34.602–89

💡 Примечание
В проектах RPA полное ТЗ по ГОСТ 34 используется редко. Чаще применяется упрощённый документ — Описание программного робота (ОПР), который включает ключевые элементы ТЗ, адаптированные под специфику роботизации (алгоритм работы, взаимодействие с ИТ-ресурсами, ограничения).

Однако знание структуры ТЗ по ГОСТ 34 важно для работы в госсекторе, крупных корпорациях и при интеграции RPA-решений в сложные ИС

15. Назначение и структура концепции и границ проекта/продукта.

Вопрос №15: «Назначение и структура концепции и границ проекта/продукта»
Ответ основан на материалах курса «Роботизация бизнес-процессов (RPA) на платформе Атом.РИТА», в частности — Лекция 3, Лекция 4, Лекция 6.

🔹 1. Назначение концепции

Концепция — это документ, формируемый на ранней стадии проекта (после предпроектного обследования), который:

Фиксирует общее видение решения проблемы или реализации возможности.
Отражает основные цели, границы, ограничения и ключевые требования к будущей системе.
Служит основой для принятия решения о целесообразности дальнейшей разработки (в том числе финансовой).
Является промежуточным звеном между потребностью бизнеса и техническим заданием.
Источник: Лекция 6, слайд 5; Лекция 3, слайд 5

✅ Основные функции концепции:

Обоснование необходимости проекта.
Определение стратегического направления.
Выбор архитектурных и технологических подходов (на высоком уровне).
Оценка рисков и ресурсов.
Подготовка к формированию ТЗ или ОПР (Описания программного робота).

🔹 2. Структура концепции (типовая)

Хотя в лекциях не приводится жёсткая форма, по аналогии с ГОСТ 34 и практикой аналитики концепция обычно включает:

Цель и задачи проекта
Анализ текущего состояния (as-is)
Предлагаемое решение (to-be)
Обоснование выбора технологии / метода (например, RPA)
Ожидаемые результаты и эффект
Ограничения и допущения
Оценка трудозатрат и стоимости

Риски и пути их минимизации
В RPA-проектах часть этой информации может быть отражена уже в Offer-card и позже развёрнута в ОПР.

🔹 3. Границы проекта (Project Scope)
Границы проекта — это чёткое определение того, что входит в проект, а что — нет. Это критически важно для управления ожиданиями заказчика и предотвращения «раздувания» требований (scope creep).

Источник: Лекция 4, слайды 10–11

✅ Элементы определения границ:
Что автоматизируется (конкретные процессы, подпроцессы).
Какие системы задействованы (СЭД, 1С, Outlook и т.д.).
Какие шаги остаются за человеком.
Какие исключения не обрабатываются роботом.
Периодичность выполнения (ежедневно, по запросу и т.п.).
Объём данных (например, до 1000 строк за раз).
Пример из RPA:

«Робот обрабатывает входящие счета в формате PDF из почтового ящика finance@company.ru, но не обрабатывает бумажные сканы без OCR и не согласует суммы выше 1 млн руб.»

🔹 4. Отличие концепции от ТЗ и ОПР
Документ
Уровень детализации
Цель
Концепция
Высокий (стратегический)
Обосновать необходимость проекта
ТЗ / ОПР
Детальный (операционный)
Описать, как именно будет работать система/робот
Источник: Лекция 3, слайд 5: «Зачем? → Что? → Как?»

🔹 Вывод
Концепция — это стратегический документ, отвечающий на вопрос «зачем и что вообще делать?».

Границы проекта — это инструмент фокусировки усилий и защиты от необоснованного расширения функционала.
В RPA-проектах концепция часто оформляется в виде Offer-card, а границы — явно прописываются в ОПР.
Как говорится в лекции:
«Без ТЗ – результат хз...» — а без чёткой концепции и границ проект может не начаться вовсе.

16. Структура текстового представления сценария использования, назначения разделов

Вопрос №16: Структура текстового представления сценария использования (Use Case), назначение разделов

Ответ основан на материалах курса «Роботизация бизнес-процессов (RPA) на платформе Атом.РИТА», в частности — Лекция 4, а также общепринятых практиках моделирования (BABOK, UML).

🔹 Что такое сценарий использования (Use Case)?

Сценарий использования — это описание последовательности действий, которые выполняет система и/или пользователь для достижения конкретной цели. В RPA он помогает чётко зафиксировать логику взаимодействия робота с ИТ-системами и пользователями.

Источник: Лекция 4, слайды 12–15

🔹 Типовая структура текстового сценария использования

Раздел
Назначение
1. Название (Имя Use Case)
Краткое, понятное название, отражающее цель (например: «Обработка входящего счёта»).
2. Краткое описание
Общее пояснение, зачем нужен этот сценарий, какую бизнес-задачу решает.
3. Акторы (Actors)
Участники сценария: кто инициирует (основной актор) и кто участвует (второстепенные акторы).

Пример: «Бухгалтер», «Программный робот», «ERP-система».
4. Предусловия (Preconditions)
Условия, которые должны быть выполнены ДО начала сценария.
Пример: «В почтовом ящике есть письмо со счётом в формате PDF».
5. Постусловия (Postconditions)
Состояние системы ПОСЛЕ успешного завершения сценария.
Пример: «Счёт зарегистрирован в ERP, письмо перемещено в папку “Обработано”».
6. Основной поток (Main Flow / Basic Path)
Пошаговое описание идеального сценария без ошибок.
Каждый шаг — одно действие (пользователя или системы).
Нумеруется: 1, 2, 3…
7. Альтернативные потоки (Alternative Flows)
Варианты отклонений от основного сценария:
— ошибки,
— исключения,
— дополнительные условия.
Пример: «Если файл не PDF → отправить уведомление и пропустить письмо».
8. Исключения (Exceptions)
Критические ошибки, при которых сценарий не может быть завершён.
Пример: «Нет доступа к ERP → остановка робота, уведомление администратора».
9. Бизнес-правила (опционально)
Ссылки на регламенты, политики, формулы, ограничения.
Пример: «Сумма свыше 1 млн руб. требует согласования».

- 🔹 Пример фрагмента (из RPA-контекста):

Название: Автоматическая загрузка счетов из почты
Акторы: Программный робот, Outlook, ERP
Предусловие: В папке «Входящие» есть письмо с темой «Счёт» и вложением .pdf
Основной поток:

Робот подключается к Outlook.
Находит письмо с вложением .pdf.

Сохраняет вложение во временную папку.
Передаёт файл в модуль OCR.
Загружает данные в ERP.
Перемещает письмо в папку «Обработано».
Альтернативный поток A1:
A1.1. Если вложений несколько → обрабатывает только первое.
A1.2. Если нет .pdf → пропускает письмо, логирует событие.

- Назначение сценария использования в RPA:

Чётко описать алгоритм работы робота.
Служить основой для Описания программного робота (ОПР).
Использоваться при тестировании (сценарии тест-кейсов).
Обеспечить согласование между владельцем процесса, ИТ и разработчиком.
Источник: Лекция 4, слайд 14: «Use Case — один из способов описания требований»

- Вывод

Текстовый сценарий использования — это структурированное, понятное и проверяемое описание поведения системы. Его чёткая структура позволяет:

избежать двусмысленности,
предусмотреть исключения,
обеспечить полноту автоматизации,
упростить передачу требований разработчику.
В проектах RPA он особенно важен, так как робот выполняет действия буквально — любая недосказанность приведёт к сбою.

17. Назначение и структуры пользовательской истории

Вопрос №17: Назначение и структура пользовательской истории (User Story)

Ответ основан на материалах курса «Роботизация бизнес-процессов (RPA) на платформе Атом.РИТА», в частности — Лекция 4, а также общепринятых Agile-практиках (Agile Manifesto, INVEST).

🔹 Назначение пользовательской истории
Пользовательская история (User Story) — это краткое, ориентированное на пользователя описание функциональности, которая приносит ценность заказчику или конечному пользователю.

Основные цели User Story:

Сфокусироваться на бизнес-ценности, а не на технической реализации.
Обеспечить гибкость в условиях меняющихся требований.
Упростить приоритизацию и планирование работ (например, в спринтах).
Служить основой для диалога между аналитиком, заказчиком и разработчиком.
Источник: Лекция 4, слайды 9–11

🔹 Стандартная структура User Story
Формат по шаблону:

«Как [роль], я хочу [цель], чтобы [выгода/бизнес-ценность]»

Пример:
«Как бухгалтер, я хочу, чтобы робот автоматически выгружал данные из PDF-счёта в ERP, чтобы сократить время обработки с 2 часов до 15 минут».

🔹 Дополнительные элементы (расширенная структура)
Хотя сама User Story — это одна строка, на практике к ней добавляются:

1. Критерии приёмки (Acceptance Criteria)
— Чёткие, проверяемые условия, при выполнении которых история считается завершённой.
Пример:

«Робот обрабатывает только PDF-файлы».
«Если файл повреждён — отправляется уведомление и письмо не удаляется».
«Данные заносятся в поля ERP: номер счёта, дата, сумма, ИНН поставщика».
2. Приоритет
— Например, по методу MoSCoW: Must, Should, Could, Won’t.

3. Оценка трудозатрат
— В story points или часах (часто делается командой на планировании).

Источник: Лекция 5 — MoSCoW; Лекция 4 — критерии приёмки

🔹 Принципы качественной User Story (INVEST)

Буква	Значение	Пояснение
I	Independent	По возможности независима от других историй
N	Negotiable	Не детализирована до мелочей — оставляет пространство для диалога
V	Valuable	Приносит ценность пользователю или бизнесу
E	Estimable	Её можно оценить по трудозатратам
S	Small	Достаточно мала для реализации за 1–2 дня
T	Testable	Есть чёткие критерии приёмки

Источник: Лекция 4, слайд 10

🔹 Применение в RPA-проектах
Хотя классические RPA-проекты часто используют Описание программного робота (ОПР), User Story применяется:

На этапе инициации и приоритизации автоматизаций.
При работе в гибридных или Agile-командах.
Для описания пользовательских потребностей, которые лежат в основе автоматизации.
Пример RPA-User Story:

«Как сотрудник отдела закупок, я хочу, чтобы робот ежедневно сверял данные в Excel и 1С, чтобы избежать расхождений в отчётах».

🔹 Вывод
Пользовательская история — это инструмент фокусировки на ценности, а не на технических деталях.
Её структура проста, но эффективна:

Формат «Как… я хочу… чтобы…» задаёт контекст,
Критерии приёмки обеспечивают тестируемость,
Принципы INVEST гарантируют качество.

В RPA она помогает правильно сформулировать цель автоматизации и согласовать её с заказчиком до начала разработки.

18. Цели приоритезации требований

Вопрос №18: Цели приоритезации требований

Ответ основан на материалах курса «Роботизация бизнес-процессов (RPA) на платформе Атом.РИТА», в частности — Лекция 5, а также общепринятых практиках управления требованиями (BABOK, Agile).

🔹 Основные цели приоритизации требований:

1. Фокусировка на ценности для бизнеса
Выделение требований, которые приносят наибольший эффект (экономию времени, снижение ошибок, рост дохода).
В RPA это особенно важно: автоматизируются в первую очередь процессы с высоким ROI и большим объёмом рутины.
Источник: Лекция 5 — расчёт эффекта в ПРМ

2. Оптимизация использования ресурсов
Разработка всех требований сразу часто невозможна из-за ограничений по:
времени,
бюджету,
кадрам.
Приоритизация позволяет максимально эффективно использовать имеющиеся ресурсы.

3. Управление сроками и поэтапной реализацией
Позволяет разбить проект на релизы или спринты.
Заказчик получает работающий функционал быстрее (MVP — минимально жизнеспособный продукт).
Источник: Лекция 5 — итеративная разработка

4. Снижение рисков
Реализация критически важных требований на ранних этапах позволяет:
проверить гипотезы,
выявить технические ограничения,
скорректировать курс до масштабных затрат.

5. Управление ожиданиями заказчика
Чёткое объяснение, почему одни функции делаются раньше других, повышает доверие.
Помогает избежать ситуации «всё сразу и бесплатно».

6. Поддержка гибкости в условиях изменений
Если требования меняются (что типично для Agile), приоритизация позволяет быстро перестроить план без потери фокуса.
7. Обоснование отказа от низкоприоритетных задач
Не все идеи стоит реализовывать. Приоритизация помогает откладывать или отклонять малоценные функции.
Источник: Лекция 5 — метод MoSCoW: «Won't have»

🔹 Пример из RPA-практики:
При наличии 20 процессов-кандидатов на автоматизацию аналитик:

оценивает их по трудозатратам и эффекту,
применяет метод MoSCoW или RICE,
выбирает 5 Must-have процессов для первого этапа,
остальные — в бэклог или на будущие этапы.

🔹 Вывод
Приоритизация требований — это не просто расстановка номеров, а стратегический инструмент управления проектом, направленный на:

максимизацию ценности,
минимизацию рисков,
рациональное использование ресурсов.

Как говорится в лекции:

«Нельзя всё автоматизировать — нужно автоматизировать самое важное».

19. Основные методы приоритезации: MoSCoW, Кано-анализ, по Вигерсу

Вопрос №19: Основные методы приоритезации — MoSCoW, Кано-анализ, по Вигерсу

Ответ основан на материалах курса «Роботизация бизнес-процессов (RPA) на платформе Атом.РИТА» (в частности, Лекция 5) и общепринятых практиках управления требованиями.

🔹 1. MoSCoW
📌 Назначение:
Метод для классификации требований по степени важности и срочности, особенно в условиях ограниченных ресурсов.

📌 Структура:
M — Must have — критически важные требования. Без них решение не имеет смысла или не будет принято.
S — Should have — важные, но не критичные. Их отсутствие ухудшает, но не блокирует решение.
C — Could have — желательные, но не обязательные. «Приятный бонус», если останутся ресурсы.
W — Won’t have (this time) — требования, которые сознательно откладываются на будущее.
Источник: Лекция 5, слайд 4

✅ Преимущества:
Простота понимания заказчиком.
Чёткое разделение «жизненно важного» и «желаемого».
Подходит для RPA-проектов при выборе процессов для автоматизации.
🔹 2. Кано-анализ (Kano Model)
📌 Назначение:
Метод для классификации требований по влиянию на удовлетворённость клиента.

📌 Категории требований:

Тип	Описание	Пример
Базовые (Must-be)	Ожидаемые по умолчанию. Их отсутствие вызывает **сильное недовольство**, но наличие — **не радует**.	Робот не теряет данные.
Ожидаемые (One-dimensional)	Чем больше — тем лучше. Удовлетворённость **прямо пропорциональна** реализации.	Чем быстрее обработка — тем выше удовлетворённость.
Восхищающие (Attractive/Delighter)	Неожиданные «фишки». Их отсутствие **не замечается**, но наличие — **вызывает восторг**.	Робот сам отправляет отчёт руководителю по окончании.
Нейтральные	Не влияют на удовлетворённость.	Цвет иконки робота.
Обратные	Вызывают раздражение, даже если реализованы.	Робот отправляет слишком много уведомлений.

Источник: Лекция 5, слайд 5

✅ Преимущества:
Помогает выйти за рамки «функционал = ценность».
Фокус на эмоциональной реакции пользователя.
Полезен при создании «умных» RPA-решений с UX-подходом.

🔹 3. Метод по Вигерсу (Wiegers’ Prioritization)

📌 Назначение:
Количественный метод приоритизации на основе балльной оценки по нескольким критериям.

📌 Формула:
Приоритет
=
Стоимость + Риск
Значимость + Стабильность
Приоритет=
Значимость + Стабильность
Стоимость + Риск

Но чаще используется упрощённая версия с весами:
Где:
✅ Преимущества:
Объективная, количественная оценка.
Учитывает стоимость реализации, а не только выгоду.
Подходит для сравнения разнородных требований (например, автоматизация vs доработка интерфейса).

🔹 Сравнение методов

🔹 Сравнение методов

Метод	Тип	Лучше подходит для	Особенность
MoSCoW	Качественный	Быстрой классификации в команде	Простота, фокус на must-have
Кано	Качественный	UX-ориентированных решений	Учёт эмоций и скрытых ожиданий
Вигерс	Количественный	Обоснования приоритетов перед руководством	Числовая оценка, учёт затрат

🔹 Применение в RPA

MoSCoW — основной метод в RPA-проектах для отбора процессов на автоматизацию.
Кано — полезен при проектировании «умных» роботов с уведомлениями, логированием, самообслуживанием.
Вигерс — применяется при наличии нескольких инициатив и необходимости обосновать ROI (например, в Offer-card).

🔹 Вывод

Выбор метода зависит от контекста:

Нужна простота и скорость → MoSCoW.
Важна удовлетворённость пользователей → Кано.
Требуется объективное сравнение → по Вигерсу.
В реальных проектах часто комбинируют методы: например, сначала MoSCoW, затем Кано для детализации «Must-have».

20. Что такое матрица трассировки и зачем она используется

Вопрос №20: Что такое матрица трассировки и зачем она используется?

Ответ основан на материалах курса «Роботизация бизнес-процессов (RPA) на платформе Атом.РИТА», в частности — Лекция 4, а также общепринятых практиках управления требованиями (BABOK, ГОСТ 34).

🔹 Что такое матрица трассировки?

Матрица трассировки требований (Requirements Traceability Matrix, RTM) — это таблица, которая устанавливает и отслеживает связи между элементами проекта на разных уровнях:

от бизнес-целей → к требованиям,
от требований → к функциональности,
от функциональности → к тест-кейсам,
от тест-кейсов → к результатам тестирования.
Источник: Лекция 4, слайд 16

🔹 Зачем используется матрица трассировки?

✅ 1. Полнота покрытия требований

Позволяет убедиться, что все требования реализованы и ничего не упущено.
Особенно важно в RPA: если шаг из ОПР не реализован — робот упадёт.

✅ 2. Избежание «лишнего» функционала

Помогает выявить функции, не связанные с бизнес-целями («золотое покрытие»).

✅ 3. Управление изменениями

При изменении требования легко найти:
какие модули затронуты,
какие тесты нужно переписать.

✅ 4. Поддержка тестирования
Гарантирует, что каждое требование покрыто тестом.
Упрощает подготовку протоколов испытаний и приёмки.
✅ 5. Соблюдение регуляторных норм
В госсекторе и финансах (ГОСТ 34, ISO) трассировка — обязательное условие документирования.
Источник: Лекция 4 — «Требования должны быть трассируемыми»

🔹 Пример из RPA
Допустим, в ОПР указано:

«Робот должен проверять наличие подписи в PDF-счете».

В матрице трассировки будет:

ссылка на бизнес-цель: «Снизить риск оплаты недействительных счетов»,
ID требования: REQ-042,
тест-кейс: TC-042 — «Проверка отсутствия подписи → отправка на ручную проверку»,
статус: «Реализовано в версии 2.1».
Если позже изменится правило (например, подпись больше не нужна), аналитик быстро найдёт, что менять в коде и тестах.

🔹 Вывод
Матрица трассировки — это инструмент контроля и прозрачности в проекте. Она:

связывает бизнес и техническую реализацию,
снижает риски упущенных или «забытых» требований,
обеспечивает качество и соответствие ожиданиям заказчика.
Как говорится в лекции:

«Хорошее требование — трассируемое требование».

21. Назначение схем «как сейчас» и «как будет», целей их использования

Вопрос №21: Назначение схем «как сейчас» (as-is) и «как будет» (to-be), цели их использования

Ответ основан на материалах курса «Роботизация бизнес-процессов (RPA) на платформе Атом.РИТА», в частности — Лекции 6, 7, 8, а также общепринятых практиках бизнес-анализа (BABOK, BPMN).

🔹 1. Схема «как сейчас» (As-Is)
📌 Назначение:

Описание текущего состояния бизнес-процесса — того, как он фактически выполняется на момент анализа.

✅ Цели использования:
Выявить проблемы, узкие места, избыточные действия (например, дублирование, ручной ввод, ожидания).
Зафиксировать реальную логику работы, включая неформальные обходные пути.
Служить базой для сравнения с будущим состоянием.
Обеспечить общее понимание процесса у всех стейкхолдеров (владельца процесса, ИТ, аналитика).
Оценить объём рутинных операций, пригодных для автоматизации.
Источник: Лекция 6 — моделирование текущего состояния

🔹 2. Схема «как будет» (To-Be)
📌 Назначение:
Описание целевого состояния процесса — того, как он должен работать после изменений (в том числе после внедрения программного робота).

✅ Цели использования:
Показать результат автоматизации или оптимизации.
Чётко определить, какие шаги остаются за человеком, а какие передаются роботу.
Согласовать с заказчиком новую логику работы до начала разработки.
Служить основой для Описания программного робота (ОПР).
Обеспечить проверяемость — можно сравнить фактический результат с моделью to-be.
Источник: Лекция 8 — проектирование целевого процесса

🔹 Сравнение и взаимосвязь

Аспект	As-Is	To-Be
Фокус	Реальность	Цель
Автоматизация	Показывает, что можно автоматизировать	Показывает, что уже автоматизировано
Роль в RPA	Исходная база для анализа	Техническое задание для разработчика
Документ	Часто используется в Offer-card	Входит в ОПР

🔹 Пример из RPA

As-Is:

Бухгалтер ежедневно:

Открывает Outlook → 2. Ищет письма со счетами → 3. Сохраняет PDF → 4. Вручную вводит данные в 1С → 5. Архивирует письмо.

To-Be:

Робот проверяет почту → 2. Извлекает PDF → 3. Через OCR распознаёт данные → 4. Загружает в 1С → 5. Перемещает письмо в папку «Обработано».

Бухгалтер только проверяет исключения.

🔹 Нотации для моделирования

BPMN — наиболее распространена в RPA-проектах.

IDEF0, eEPC, UML Activity Diagram — также допустимы.

Источник: Лекции 6–8

🔹 Вывод

Схемы «as-is» и «to-be» — это основа аналитической работы в RPA:

As-is отвечает на вопрос: «Что мы автоматизируем?»

To-be отвечает на вопрос: «Как это будет работать после автоматизации?»

Их совместное использование позволяет:

избежать автоматизации неэффективных процессов,

чётко разделить зоны ответственности человека и робота,

обеспечить прозрачность и согласованность между бизнесом и ИТ.

22. Основные нотаций и диаграмм для моделирования бизнес-процессов

Вопрос №22: Основные нотации и диаграммы для моделирования бизнес-процессов

Ответ основан на материалах курса «Роботизация бизнес-процессов (RPA) на платформе Атом.РИТА», в частности — Лекции 6, 7, 8, а также общепринятых стандартах бизнес-моделирования.

🔹 Основные нотации для моделирования бизнес-процессов

1. BPMN (Business Process Model and Notation)
Наиболее распространённая нотация в современных проектах, включая RPA.
Поддерживает детальное описание последовательности шагов, параллельных потоков, условий, исключений.
Использует понятные графические элементы:
Задачи (Task),
События (Start/End/Intermediate),
Шлюзы (Gateway — для ветвлений),
Пулы и дорожки (Pool/Lane — для ролей/систем).
Преимущество: подходит как для аналитиков, так и для разработчиков; легко читается заказчиком.
Источник: Лекция 6, слайд 3

2. IDEF0 (Integration Definition for Function Modeling)
Функционально-ориентированная нотация.
Описывает что делает система, а не как.
Использует блоки с входами, выходами, механизмами и управлениями (вход → функция → выход + управление + ресурсы).
Хорошо подходит для высокоуровневого анализа и декомпозиции сложных процессов.
Часто применяется в госсекторе и при работе по ГОСТ.
Источник: Лекция 7, слайд 3

3. eEPC (extended Event-driven Process Chain)
Немецкая нотация, популярна в ERP-средах (SAP).
Чередует события (Event) и функции (Function).
Поддерживает логические операторы: И, ИЛИ, XOR.
Удобна для описания триггерных процессов («когда произошло событие X — запускается функция Y»).
Источник: Лекция 8, слайд 3

4. UML Activity Diagram (Диаграмма активности UML)
Часть Unified Modeling Language.
Похожа на BPMN, но изначально ориентирована на описание логики программного обеспечения.
Используется, когда процесс тесно связан с поведением системы.
Поддерживает параллелизм, условия, исключения.
Источник: Лекция 6, слайд 5

🔹 Сравнение нотаций

Нотация	Основной фокус	Преимущества	Типичное применен
BPMN	Последовательность действий	Визуальная ясность, поддержка исключений, широкое принятие	RPA, корпоративные процессы
IDEF0	Функции и потоки данных	Строгая структура, хороша для анализа «чёрного ящика»	Госпроекты, стратегический анализ
eEPC	События → действия	Интеграция с SAP, чёткая логика триггеров	ERP-автоматизация
UML Activity	Логика выполнения	Интеграция с разработкой ПО	IT-проекты, системный анализ

🔹 Какую нотацию выбрать в RPA?
Рекомендуется BPMN — она:
интуитивно понятна,
поддерживает детализацию до уровня кликов (важно для робота),
легко согласуется с владельцем процесса.
IDEF0 — полезна на этапе предпроектного обследования для высокоуровневого взгляда.
eEPC/UML — применяются при наличии корпоративных стандартов или интеграции с определёнными системами.
Источник: Лекции 6–8 — рекомендации по выбору нотации

🔹 Вывод
Моделирование бизнес-процессов — ключевой этап в RPA.
Выбор нотации зависит от:

аудитории (бизнес vs ИТ),
уровня детализации,
корпоративных стандартов.
BPMN — наиболее универсальный и практичный выбор для большинства RPA-проектов, так как позволяет точно описать последовательность действий, которые должен выполнять программный робот.

Вопрос №23: Нотация UML и основные её диаграммы, полезные бизнес-аналитику

🔹 Что такое UML?
UML (Unified Modeling Language) — это стандартизированный язык графического моделирования, предназначенный для визуального описания структуры и поведения программных систем.

Хотя UML изначально ориентирован на разработчиков ПО, ряд его диаграмм активно используются бизнес-аналитиками для:

описания логики процессов,

согласования с заказчиком,
передачи требований разработчикам.
Источник: Лекция 6, слайд 5 — упоминание UML Activity Diagram как альтернативы BPMN

🔹 Основные диаграммы UML, полезные бизнес-аналитику
1. Диаграмма активности (Activity Diagram)
Аналог: BPMN.
Назначение: Моделирование последовательности действий, ветвлений, параллельных потоков.
Элементы:
Начало/конец,
Действия (прямоугольники),
Решения (ромбы),
Параллельные потоки («вилки» и «слияния»).
Применение в RPA:
— Описание алгоритма работы робота,
— Визуализация «as-is» и «to-be» процессов.
✅ Почему полезна: интуитивно понятна заказчику, поддерживает исключения и условия.

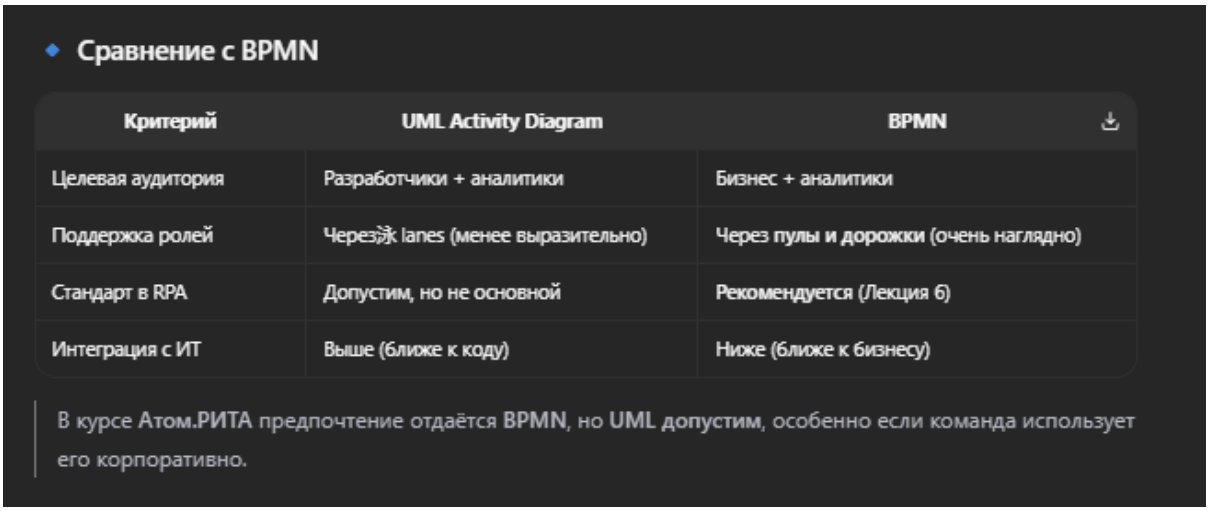
2. Диаграмма вариантов использования (Use Case Diagram)
Назначение: Отображение взаимодействия акторов (ролей) с функциональностью системы.
Элементы:
Акторы (человечки),
Use Cases (овалы),
Связи (линии),
Отношения «include» / «extend».
Применение:
— Быстрое согласование границ системы,
— Определение ключевых функций на раннем этапе.
✅ Почему полезна: помогает ответить на вопрос: «Кто что делает в системе?»

Пример:
Актор «Бухгалтер» → Use Case «Загрузить счёт в ERP»
Актор «Программный робот» → Use Case «Обработать входящие письма»

3. Диаграмма последовательности (Sequence Diagram)
Назначение: Моделирование взаимодействия объектов во времени — кто, когда и что вызывает.
Элементы:
Жизненные линии (вертикальные линии),
Сообщения (горизонтальные стрелки),
Активации (прямоугольники на линиях).
Применение в RPA:
— Описание взаимодействия робота с API, почтой, базой данных,
— Уточнение порядка вызовов при интеграции.

✅ Почему полезна: показывает хронологию событий, критичную для автоматизации.

4. Диаграмма классов (Class Diagram) — опционально
Назначение: Описание структур данных — сущностей, их атрибутов и связей.
Применение:
— При проектировании хранилища данных для робота,
— При работе с JSON/XML-структурами,
— При описании справочников (например, «Клиент», «Счёт», «Поставщик»).
⚠️ Менее востребована у бизнес-аналитика, но полезна при тесной работе с разработчиками.



🔹 Сравнение с BPMN

Критерий	UML Activity Diagram	BPMN
Целевая аудитория	Разработчики + аналитики	Бизнес + аналитики
Поддержка ролей	Через lanes (менее выразительно)	Через пулы и дорожки (очень наглядно)
Стандарт в RPA	Допустим, но не основной	Рекомендуется (Лекция 6)
Интеграция с ИТ	Выше (ближе к коду)	Ниже (ближе к бизнесу)

> В курсе Атом.РИТА предпочтение отдаётся BPMN, но UML допустим, особенно если команда использует его корпоративно.

🔹 Вывод
Для бизнес-аналитика наиболее полезны следующие диаграммы UML:

Activity Diagram — для описания логики процессов (альтернатива BPMN),
Use Case Diagram — для фиксации границ системы и ролей,
Sequence Diagram — для детализации взаимодействий (особенно при API-интеграциях).
Хотя в RPA-проектах чаще используется BPMN, знание UML расширяет инструментарий аналитика и улучшает взаимодействие с ИТ-командой.

24. Виды и цели презентаций и демонстраций решений

🔹 Общая цель презентаций и демонстраций
Презентации и демонстрации решений направлены на:

передачу информации заинтересованным лицам,
получение обратной связи,
согласование или утверждение решения,
повышение вовлечённости и доверия к проекту.

🔹 Основные виды презентаций и демонстраций в RPA-проектах
1. Презентация идеи / Offer-card
Когда: На этапе инициации.
Аудитория: ЛПР, владелец процесса, команда RPA.
Цель:

— Обосновать целесообразность автоматизации,
— Показать потенциальный эффект (в ПРМ, рублях),
— Получить одобрение на запуск аналитики.
Формат: Краткая презентация (3–5 слайдов): проблема → решение → эффект.
Источник: Лекция 11

2. Демонстрация модели «as-is» и «to-be»
Когда: После аналитической фазы.
Аудитория: Владелец процесса, владелец ИТ-ресурса.
Цель:
— Согласовать текущее состояние,
— Утвердить целевую модель автоматизации,
— Выявить упущенные шаги или исключения.
Формат: BPMN-диаграммы + пояснения.
Источник: Лекции 6–8

3. Презентация Описания программного робота (ОПР)
Когда: Перед передачей в разработку.
Аудитория: Разработчик, владелец ИТ-ресурса, служба ИБ.
Цель:
— Чётко донести алгоритм работы робота,
— Согласовать технические ограничения,
— Получить ЛИ (листы исполнения).
Формат: Документ + устное пояснение ключевых моментов.
Источник: Лекция 11

4. Демонстрация работающего прототипа / MVP
Когда: По завершении разработки (тестовая среда).
Аудитория: Заказчик, владелец процесса.
Цель:
— Показать реальную работу робота,
— Получить обратную связь до внедрения,
— Подтвердить соответствие ОПР.
Формат: Живая демонстрация на тестовых данных.
Источник: Лекция 11 — совместное тестирование

5. Презентация результатов / демо-сессия (Review)
Когда: После внедрения или в конце спринта (в Agile).
Аудитория: Команда, заказчик, ЛПР.
Цель:
— Продемонстрировать достигнутый эффект,
— Подтвердить выполнение KPI,
— Обсудить дальнейшие шаги (доработки, масштабирование).
Формат: Слайды + метрики + живая демонстрация.
Источник: Лекция 5 — итеративная разработка

6. Обучающая демонстрация для конечных пользователей
Когда: При запуске в продуктив.

Аудитория: Сотрудники, чья работа затрагивается.
Цель:
— Обучить работе с новым процессом,
— Объяснить, что изменилось,
— Снизить сопротивление изменениям.
Формат: Инструкция + короткий видеоролик или live-демо.

🔹 **Типология по целям (обобщённо)**

Тип	Цель
Убеждающая	Получить одобрение, бюджет, поддержку (Offer-card).
Согласующая	Утвердить требования, модель, технические детали (ОПР, as-is/to-be).
Демонстрационная	Показать работающее решение (MVP, Review).
Обучающая	Научить пользоваться (инструктаж пользователей).

🔹 Вывод
Презентации и демонстрации — не просто «показ картинок», а ключевые точки взаимодействия с заинтересованными лицами на всех этапах RPA-проекта.
Их правильное использование позволяет:

избежать недопонимания,
ускорить согласования,
повысить качество решения,
обеспечить успешное внедрение.
Как говорится в лекциях:

«Аналитик — связующее звено между Клиентом и Разработчиком» — а презентации и демо — его основные инструменты коммуникации.

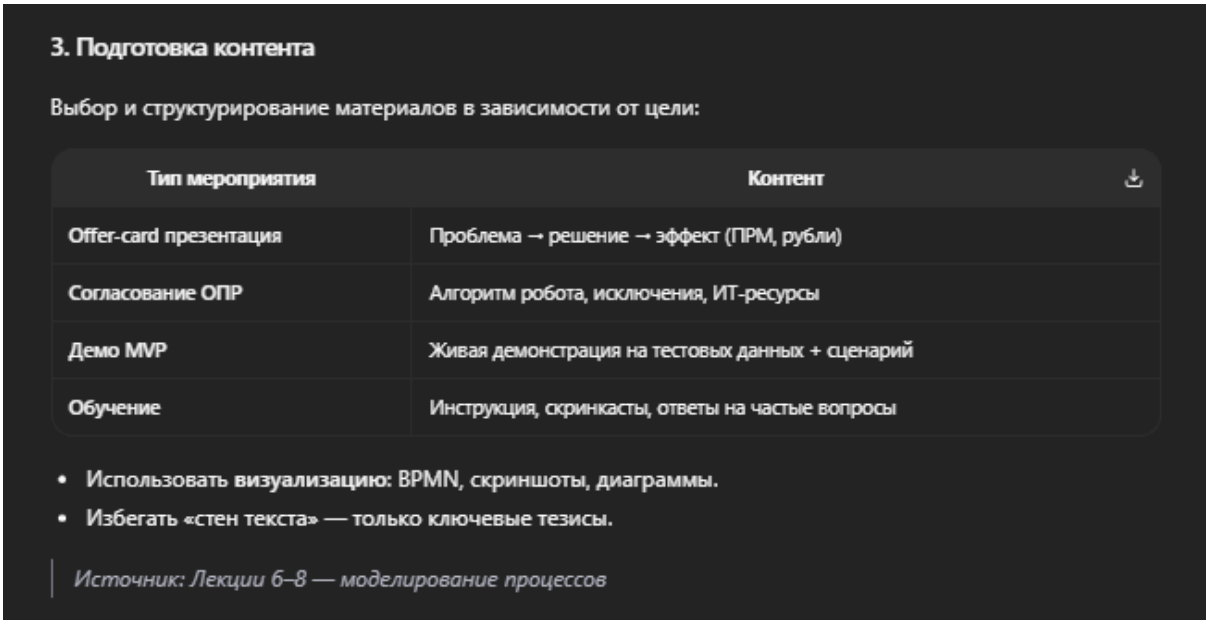
25. Этапы подготовки к проведению презентаций и демонстраций

🔹 Основные этапы подготовки
1. Определение цели презентации/демонстрации
Чётко сформулировать, зачем проводится встреча:
Получить одобрение?
Согласовать требования?
Продемонстрировать работающий робот?
Обучить пользователей?
Цель определяет формат, содержание и аудиторию.
Источник: Лекция 5 — SMART-цели для встреч

2. Идентификация аудитории
Кто будет присутствовать?
ЛПР, владелец процесса, разработчик, конечный пользователь?
От этого зависит:

Уровень детализации,
Язык изложения (бизнес vs технический),
Акценты (эффект vs реализация).
Источник: Лекция 2 — матрица стейкхолдеров

3. Подготовка контента
Выбор и структурирование материалов в зависимости от цели:



3. Подготовка контента

Выбор и структурирование материалов в зависимости от цели:

Тип мероприятия	Контент
Offer-card презентация	Проблема → решение → эффект (ПРМ, рубли)
Согласование ОПР	Алгоритм робота, исключения, ИТ-ресурсы
Демо MVP	Живая демонстрация на тестовых данных + сценарий
Обучение	Инструкция, скринкасты, ответы на частые вопросы

- Использовать **визуализацию:** BPMN, скриншоты, диаграммы.
- Избегать «стен текста» — только ключевые тезисы.

> *Источник: Лекции 6–8 — моделирование процессов*

4. Техническая подготовка
Проверить:
Доступ к демонстрационной среде,
Работоспособность робота (на тестовых данных!),
Стабильность интернета, проектора, звука,
Резервные материалы (офлайн-видео, PDF).
Для RPA-демо: подготовить изолированный тестовый сценарий, чтобы избежать сбоев в продакшене.
Источник: Лекция 11 — тестирование перед внедрением

5. Репетиция и тайминг
Проговорить выступление вслух.
Уложиться в отведённое время (обычно 10–30 мин).
Продумать ответы на возможные вопросы и возражения:
«А если система обновится?»
«Как обрабатываются ошибки?»
«Почему не автоматизировать ещё и это?»
6. Подготовка материалов для участников
Разослать повестку дня заранее.
Подготовить раздаточные материалы:
PDF-презентацию,
ОПР,
Инструкцию,
Форму обратной связи.

В RPA-проектах часто прикладывают листы исполнения (ЛИ) для подписания.
Источник: Лекция 11 — документы для согласования

7. Определение формата взаимодействия
Будет ли это:
Очное совещание?
Видеоконференция?
Самостоятельный просмотр записи?
Настроить инструменты: Zoom, Teams, Miro, YouGile и т.д.
🔹 Дополнительно: Agile-подход (если применяется)
В Scrum: демо проводится на Sprint Review.
В Kanban: демо — по готовности фичи.
Всегда включает обратную связь и обсуждение следующих шагов.
Источник: Лекция 5 — итеративная разработка

🔹 Вывод
Подготовка к презентации или демонстрации — это не просто «собрать слайды», а стратегический процесс, включающий:

Чёткую цель,
Понимание аудитории,
Релевантный контент,
Техническую надёжность,
Готовность к диалогу.
В RPA-проектах особенно важно демонстрировать работающий прототип, а не только схемы — это повышает доверие заказчика и снижает риски недопонимания.

Как говорится в лекциях:
«Без ТЗ – результат хз...», но даже с ТЗ — без хорошей демонстрации — результат может не быть принят.

26. Опишите своими словами понятие RPA.

RPA (Robotic Process Automation — роботизация бизнес-процессов) — это технология автоматизации рутинных, повторяющихся задач, которые ранее выполнял человек в цифровой среде.

Суть RPA в том, что программный робот (или «цифровой сотрудник») имитирует действия человека на уровне пользовательского интерфейса:
— открывает приложения (например, Excel, 1С, почту, браузер),
— вводит данные,
— копирует и переносит информацию между системами,
— нажимает кнопки,
— обрабатывает файлы,
— принимает простые решения на основе правил.

Робот работает точно по заданному алгоритму, без усталости, ошибок и выходных. Он не заменяет сложное мышление человека, но берёт на себя монотонную,

предсказуемую работу, освобождая сотрудников для более творческих и стратегических задач.

В отличие от классической ИТ-автоматизации (которая требует изменения кода систем или интеграции через API), RPA не вмешивается во внутреннюю архитектуру программ — он работает поверх них, как обычный пользователь. Это делает RPA быстрым, недорогим и гибким инструментом, особенно там, где нет возможности или ресурсов на глубокую интеграцию.

В контексте курса «Роботизация бизнес-процессов на платформе Атом.РИТА»:
RPA — это не про физических роботов, а про программные решения, создаваемые на основе чётко описанного бизнес-процесса (в ОПР), согласованного с владельцем процесса и ИТ-ресурса, и приносящие измеримый эффект (в ПРМ и рублях).

Пример:
Вместо того чтобы бухгалтер ежедневно вручную выгружал данные из 50 PDF-счетов в Excel и 1С, робот делает это за 10 минут — без ошибок и перерывов.

Таким образом, RPA — это мост между бизнесом и ИТ, позволяющий быстро и безопасно автоматизировать то, что можно описать по шагам.

27. Приведите 3 примера того, в каких областях могут применять роботы.

Вот три типичные области применения программных роботов (RPA) с конкретными примерами:

1. Финансы и бухгалтерия
Пример: Автоматическая обработка входящих счетов.
Робот проверяет почту, извлекает PDF-счета, распознаёт данные через OCR, сверяет их с заказами в ERP-системе (например, 1С), создаёт проводки и отправляет документы на согласование при превышении лимита.
Эффект: Сокращение времени обработки с 2 часов до 15 минут в день, снижение ошибок.
Источник: Лекция 11 — примеры автоматизации в RPA

2. Управление персоналом (HR)
Пример: Онбординг новых сотрудников.
Робот получает данные из системы найма, создаёт учётные записи в Active Directory, Outlook, корпоративном портале, отправляет приветственное письмо с инструкциями и уведомляет руководителя.
Эффект: Снижение административной нагрузки на HR, ускорение запуска сотрудника в работу.
3. IT-поддержка и управление инцидентами
Пример: Обработка заявок из Service Desk.
Робот мониторит почтовый ящик или систему типа Jira, классифицирует запросы по шаблонам («забыл пароль», «не работает принтер»), создаёт тикеты с нужными

метками, направляет их в соответствующую очередь и отправляет пользователю стандартный ответ.
Эффект: Сокращение времени реакции, разгрузка техподдержки от рутины.
💡 Дополнительно (из курса):
Как указано в Лекции 1, RPA особенно эффективен там, где процессы:

Правилозависимые (чёткий алгоритм),
Повторяющиеся (выполняются регулярно),
Структурированные (работа с формами, таблицами, интерфейсами),
Объёмные (много однотипных операций).
Такие процессы встречаются практически в любой сфере: здравоохранение (регистрация пациентов), логистика (обработка накладных), страхование (расчёт премий), госсектор (приём заявлений) и др.

Главное — точное описание процесса (в ОПР) и согласование с владельцами систем.

28. Приведите 3 примера того, какие функции могут выполнять современные роботы.

Современные программные роботы (RPA-боты) — это не просто «макросы», а интеллектуальные цифровые сотрудники, способные выполнять широкий спектр задач. Вот три ключевых примера функций:

1. Работа с данными между системами (интеграция на уровне UI)
Функция: Автоматический перенос данных из одной системы в другую без участия человека.
Пример:
Робот ежедневно:
открывает Excel-файл с отчётом продаж,
копирует данные по клиентам,
заходит в CRM (например, Битрикс24),
создаёт или обновляет карточки клиентов,
логирует результат в Google Таблицу.
Особенность: Не требует API — работает через интерфейс, как человек.
Источник: Лекция 11 — взаимодействие с ИТ-ресурсами

2. Обработка документов с использованием ИИ (OCR + NLP)
Функция: Распознавание и структурирование информации из неструктурированных документов.
Пример:
Робот:
получает сканы договоров по email,
использует OCR (например, ABBYY, Tesseract) для распознавания текста,
применяет правила или ML-модель для извлечения полей (номер договора, дата, сумма, стороны),
заносит данные в базу и отправляет уведомление юристу при несоответствии шаблону.
Особенность: Комбинация RPA + ИИ (интеллектуальная автоматизация).
Источник: Лекции 1–2 — ограничения и возможности RPA

3. Мониторинг и реагирование на события в реальном времени
Функция: Непрерывный контроль за системами и автоматическое выполнение действий по триггеру.
Пример:
Робот:
каждые 5 минут проверяет очередь заявок в Service Desk,
если появляется заявка с тегом «Критично»,
немедленно отправляет SMS-уведомление ответственному инженеру,
создаёт резервную копию логов и запускает диагностический скрипт.
Особенность: Робот работает 24/7, обеспечивая оперативность.
💡 Дополнительно:
Современные RPA-платформы (включая Атом.РИТА) поддерживают:

работу с электронной почтой, веб-сайтами, офисными приложениями,
взаимодействие через API (расширяя возможности UI-автоматизации),
логирование, обработку исключений, уведомления,
запуск по расписанию или по событию.
Главное ограничение: робот выполняет чётко описанные действия. Он не заменяет мышление, но идеально справляется с предсказуемой рутиной.

29. Какие преимущества роботизации для компании вы можете назвать

🔹 Основные преимущества роботизации (RPA) для компании:
1. Снижение операционных затрат
Роботы работают 24/7 без зарплаты, отпусков и больничных.
Замена рутинного труда человека на автоматизацию позволяет сократить FTE (Full-Time Equivalent) или перераспределить сотрудников на более ценные задачи.
Пример: автоматизация обработки 1000 счетов в день вместо найма дополнительного бухгалтера.
Источник: Лекция 11 — расчёт эффекта в ПРМ (полных рабочих месяцах)

2. Повышение точности и снижение ошибок
Робот выполняет одни и те же действия строго по алгоритму, без усталости и рассеянности.
Исключаются человеческие ошибки: опечатки, пропущенные строки, неправильные формулы.
Это критично в финансах, логистике, HR, где ошибка = штраф или репутационный ущерб.
3. Ускорение выполнения процессов
Робот работает в разы быстрее человека: например, обрабатывает документ за секунды вместо минут.
Сокращаются циклы обработки (например, время закрытия месяца, оформления заказа).
Повышается скорость реакции на запросы клиентов.
4. Масштабируемость
При росте объёмов достаточно запустить ещё один экземпляр робота — не нужно нанимать и обучать персонал.

Особенно полезно при сезонных пиках (отчётность, распродажи, налоговые декларации).
5. Улучшение соответствия требованиям (Compliance)
Все действия робота логируются, что обеспечивает:
полную аудиторскую прозрачность,
соответствие регуляторным нормам (ФЗ-152, GDPR, PCI DSS и др.).
Нет «серых зон»: если робот делает — значит, это разрешено и задокументировано.
6. Повышение удовлетворённости сотрудников
Люди освобождаются от монотонной рутины и могут заниматься аналитикой, творчеством, взаимодействием с клиентами.
Снижается выгорание и текучесть кадров в рутинных ролях.
7. Быстрый ROI (возврат инвестиций)
Внедрение RPA часто не требует изменений в ИТ-инфраструктуре (работает поверх существующих систем).
Срок окупаемости — от нескольких недель до 6 месяцев.
Эффект легко измеряется в ПРМ и рублях.
Источник: Лекция 11 — экономическое обоснование автоматизации

8. Гибкость и низкий порог входа
Не нужны глубокие интеграции через API — достаточно доступа к UI.
Подходит даже для устаревших систем (legacy), где нет возможности доработки.

🔹 Вывод

Роботизация — это не просто «технологическая игрушка», а стратегический инструмент повышения эффективности, который даёт компании:

экономию,
точность,
скорость,
управляемость.
Как говорится в лекциях:

«Автоматизировать нужно не всё подряд, а то, что приносит измеримый эффект».

Именно поэтому в RPA-проектах так важны Offer-card, ОПР и расчёт ПРМ — чтобы фокусироваться на действительно ценной автоматизации.

30. Какие недостатки роботизации для компании вы можете назвать.

Хотя роботизация (RPA) приносит значительные преимущества, у неё есть и существенные ограничения и риски, особенно при некорректном внедрении. Ниже — ключевые недостатки и вызовы.

🔹 1. Зависимость от стабильности интерфейсов

Роботы работают на уровне пользовательского интерфейса (UI).
Любое изменение в системе — обновление 1С, смена дизайна веб-сайта, переименование кнопки — ломает сценарий.
Требуется постоянное сопровождение и оперативная корректировка роботов.
Источник: Лекция 11 — необходимость Службы поддержки ПР (СППР)

🔹 2. Ограниченная «интеллектуальность»
Классический RPA-робот не принимает решений в условиях неопределённости.
Он работает только по чётким правилам: «если А — то Б».
Не справляется с:
неструктурированными данными без OCR/NLP,
вариативными форматами документов,
исключениями, не описанными в ОПР.
Источник: Лекция 1 — RPA ≠ ИИ

🔹 3. Риск автоматизации неэффективных процессов
Если автоматизировать «кривой» процесс, он станет быстрее, но всё так же кривым.
Пример: ручной дублирующий ввод данных между системами → робот будет делать это быстро, но проблема архитектуры останется.
Решение: сначала оптимизировать процесс (модель «to-be»), потом автоматизировать.
Источник: Лекции 6–8 — моделирование as-is / to-be

🔹 4. Сопротивление сотрудников и организационные барьеры
Сотрудники могут воспринимать роботов как угрозу увольнения.
Возникает скрытое саботаж: намеренное искажение данных, отказ от согласования.
Требуется change management: обучение, коммуникация, вовлечение.
🔹 5. Проблемы информационной безопасности
Роботы часто требуют учётных записей с высокими привилегиями.
Хранение логинов/паролей, доступ к конфиденциальным данным — риск утечки.
Необходимо строгое соблюдение политик ИБ, использование менеджеров паролей, аудит действий.
Источник: Лекция 11 — роль администратора ИБ ИТ-ресурса

🔹 6. Скрытые затраты на сопровождение
На первый взгляд RPA — дешёвое решение, но:
нужны аналитики, разработчики, поддержка,
требуется мониторинг, логирование, обработка инцидентов.
Без должного сопровождения роботы «умирают» через несколько месяцев.
🔹 7. Ограниченная применимость
RPA эффективен только для правилозависимых, повторяющихся, структурированных задач.
Не подходит для:
творческих задач,
работы с эмоциями (клиентская поддержка),
принятия стратегических решений.
🔹 Вывод
RPA — мощный инструмент, но не панацея. Его успешное внедрение требует:

тщательного отбора процессов,
качественного анализа (ОПР),
учёта рисков ИБ и изменений в системах,
управления изменениями в коллективе.
Как говорится в лекциях:

«Автоматизировать нужно не всё, а только то, что действительно приносит эффект и устойчиво к изменениям».

Без этого роботизация может привести к потере времени, денег и доверия к технологии.

31. Придумайте 1 пример использования технологии RPA и опишите его: пример роботизированного процесса, плюсы и минусы, какое рабочее время сотрудников он потенциально может высвободить.

📌 Пример роботизированного процесса:
Автоматическая сверка банковских выписок с внутренними платёжными поручениями

🔹 Описание процесса «как сейчас» (as-is):
Каждый рабочий день бухгалтер:

Заходит в интернет-банк и выгружает выписку за предыдущий день в формате Excel.
Открывает внутреннюю систему учёта (например, 1С) и выгружает список платёжных поручений за тот же период.
Вручную сопоставляет каждую строку из банковской выписки с платёжным поручением по дате, сумме и ИНН контрагента.
Выделяет несоответствия (например, оплата прошла, но в 1С не отражена) и отправляет уведомление ответственному сотруднику.
Заносит результаты сверки в журнал учёта.
Процесс занимает 1,5–2 часа ежедневно, подвержен ошибкам из-за усталости и монотонности.

🤖 Как работает робот (to-be):
Робот по расписанию (в 8:00 утра) подключается к интернет-банку через UI.
Скачивает выписку за вчерашний день.
Подключается к 1С, выгружает платёжные поручения.
Автоматически сопоставляет записи по ключевым полям.
Формирует отчёт о расхождениях и отправляет его на email главному бухгалтеру.
Логирует все действия в служебный журнал.
Робот выполняет задачу за 10–15 минут, без участия человека.

✅ Плюсы автоматизации:
Экономия времени: ~1,5 часа × 22 рабочих дня = 33 часа в месяц (~4 рабочих дня).
Снижение ошибок: исключены пропуски и опечатки при ручном сопоставлении.
Оперативность: сверка готова к началу рабочего дня.
Аудит: полная история действий робота — легко проверить.
Масштабируемость: при увеличении объёма операций время выполнения почти не растёт.
❌ Минусы и риски:
Зависимость от интерфейса банка: если банк обновит сайт — робот может сломаться.
Требуется доступ к 1С и интернет-банку: нужны учётные данные и согласование с ИБ.

Не обрабатывает сложные исключения: например, если один платёж разбит на несколько проводок — потребуется доработка логики.
Нужно сопровождение: при изменениях в форматах выписок — корректировка сценария.
⏱️ Потенциально высвобождаемое рабочее время:
~33 часа в месяц (примерно 4 полных рабочих дня).
Это время бухгалтер может направить на:
анализ причин расхождений,
взаимодействие с контрагентами,
участие в налоговой оптимизации,
обучение.
При масштабировании на несколько компаний или филиалов эффект умножается.

💡 Вывод
Данный пример демонстрирует классический кейс RPA:
— правилозависимый,
— повторяющийся,
— объёмный,
— работа с UI и файлами.

Такой процесс идеально подходит для роботизации и приносит быстрый, измеримый эффект — как в экономии времени, так и в повышении качества контроля.

32. Перечислите цели предпроектного обследования

🔹 Основные цели предпроектного обследования:
1. Выявление и формулировка бизнес-проблемы или потребности
Понять, что именно не устраивает заказчика: рутина, ошибки, задержки, избыточные затраты.
Ответить на вопрос: «Почему мы хотим автоматизировать?»
Источник: Лекция 1 — постановка задачи

2. Оценка целесообразности автоматизации
Определить, можно ли и стоит ли автоматизировать процесс с помощью RPA.
Проверить наличие критериев «автоматизируемости»:
повторяемость,
правила вместо суждений,
работа с цифровыми системами.
Источник: Лекция 1 — признаки подходящего процесса для RPA

3. Сбор первичной информации о текущем состоянии (as-is)
Зафиксировать, как процесс выполняется сейчас:
какие системы используются,
сколько времени занимает,
кто участвует,
где возникают ошибки и узкие места.
Источник: Лекция 6 — моделирование as-is

4. Идентификация заинтересованных лиц (стейкхолдеров)

Определить:
владельца процесса,
владельца ИТ-ресурса,
конечных пользователей,
ЛПР.
Это необходимо для дальнейшего согласования и сбора требований.
Источник: Лекция 2 — роли в RPA-проекте

5. Оценка потенциального эффекта от автоматизации
Рассчитать предварительный экономический эффект:
трудозатраты до/после,
высвобождаемое время,
снижение ошибок.
Выразить эффект в ПРМ (полных рабочих месяцах) и рублях.
Источник: Лекция 11 — расчёт эффекта

6. Формирование Offer-card (инициации идеи)
На основе обследования подготовить краткое обоснование для запуска проекта:
проблема,
предложенное решение,
ожидаемый результат.
Источник: Лекция 11 — этап инициации

7. Выявление технических и организационных ограничений
Проверить:
доступны ли системы для робота,
есть ли CAPTCHA, двухфакторная аутентификация,
разрешена ли автоматизация политикой ИБ.

- Вывод

Предпроектное обследование — это фундамент всего RPA-проекта. Его цели направлены на:

понимание сути проблемы,
проверку возможности решения через RPA,
оценку выгоды,
подготовку основания для запуска аналитики.
Без качественного обследования велик риск автоматизировать неподходящий процесс, что приведёт к потере ресурсов и недоверию к технологии.

33. Приведите 3 примера бизнес-требований.

Вот 3 примера бизнес-требований — то есть формулировок, отвечающих на вопрос «зачем?» и описывающих цели и ценность для бизнеса, а не техническую реализацию:

1. Сократить время обработки входящих счетов с 2 часов до 15 минут в день, чтобы бухгалтер мог сосредоточиться на анализе расходов, а не на рутинном вводе данных.
2. Уменьшить количество ошибок при оформлении заказов на 90%, чтобы снизить число возвратов и повысить удовлетворённость клиентов.

3. Обеспечить ежедневную актуализацию данных о наличии товаров в интернет-магазине, чтобы клиенты не оформляли заказы на отсутствующие позиции и не теряли доверие к сервису.
💡 Важно: бизнес-требования всегда связаны с измеримым эффектом (время, деньги, качество, удовлетворённость) и стратегической целью, а не с функциональностью системы.
Они лежат в основе Offer-card и последующего Описания программного робота (ОПР) в RPA-проектах.

34. Приведите 3 примера функциональных требований (включая минимум по 1 требованию Use Cases и User story).

Вот три примера функциональных требований — два в формате Use Case и один в формате User Story, как того требует условие (минимум по одному каждого типа; третий можно выбрать — здесь добавлен второй Use Case для полноты).

✅ 1. Use Case: Обработка входящего счёта из электронной почты
Название: Автоматическая обработка PDF-счёта из корпоративной почты
Акторы: Программный робот, Microsoft Outlook, ERP-система
Предусловие: В папке «Входящие» есть письмо с темой «Счёт» и вложением в формате PDF
Основной поток:
Робот подключается к Outlook.
Находит письмо с вложением .pdf.
Сохраняет файл во временную папку.
Передаёт файл модулю OCR для распознавания.
Извлекает данные: номер счёта, дата, сумма, ИНН поставщика.
Загружает данные в ERP-систему.
Перемещает письмо в папку «Обработано».
Постусловие: Счёт зарегистрирован в ERP, письмо архивировано.
✅ 2. Use Case: Формирование ежедневного отчёта по продажам
Название: Генерация и отправка сводного отчёта о продажах
Акторы: Программный робот, Excel, Power BI, Email
Предусловие: В базе данных за предыдущий день есть данные о продажах
Основной поток:
Робот подключается к источнику данных.
Выгружает информацию о продажах за вчерашний день.
Формирует Excel-отчёт с разбивкой по регионам и категориям.
Сохраняет файл на сетевой диск.
Отправляет email с отчётом руководителю отдела продаж.
Исключение: Если данных нет — отправляется уведомление «Продаж за вчера не зафиксировано».
✅ 3. User Story: Уведомление о низком остатке товара
Как менеджер по логистике,
я хочу, чтобы система автоматически проверяла остатки товаров на складе каждое утро,
чтобы я мог оперативно оформлять заказы поставщикам и избежать дефицита.

Критерии приёмки:
Проверка запускается ежедневно в 8:00.
Если остаток < порогового значения — создаётся задача в системе управления складом.
Менеджер получает email со списком товаров с низким остатком.
💡 Пояснение:
Use Case — структурированное описание взаимодействия между акторами и системой, подходит для технической передачи в разработку.
User Story — ориентирована на ценность для пользователя, используется в Agile и для быстрого согласования целей.
Оба формата являются функциональными требованиями, так как описывают что система должна делать.

35. Приведите 3 примера нефункциональных требований на выбор для одной из вымышленных разрабатываемых систем

Конечно! Ниже приведены по 3 примера нефункциональных требований для каждой из трёх указанных вымышленных систем. Нефункциональные требования описывают качественные характеристики системы: безопасность, производительность, надёжность, удобство и т.д.

🌍 1. Интернет-магазин по продаже туристических путёвок
Производительность:
Страница поиска туров должна загружаться не дольше 2 секунд даже при одновременном запросе от 5 000 пользователей (например, в период горящих предложений).
Безопасность платежей:
Все транзакции должны соответствовать стандарту PCI DSS, а данные банковских карт не храниться на серверах магазина — оплата осуществляется только через сертифицированные платёжные шлюзы (СБП, ЮKassa и др.).
Доступность и отказоустойчивость:
Сайт должен быть доступен 99,5% времени в месяц (максимум ~22 минуты простоя).
При падении основного сервера резервная система должна активироваться в течение 5 минут без потери данных о бронировании.
🐾 2. Интернет-магазин по продаже товаров для животных
Мобильная адаптивность:
Интерфейс должен корректно отображаться и функционировать на всех популярных мобильных устройствах (iOS, Android) с разрешением от 320px и выше (responsive design).
Скорость обработки заказа:
После подтверждения оплаты заказ должен автоматически передаваться в систему складского учёта в течение 10 секунд, чтобы обеспечить своевременную комплектацию.
Масштабируемость каталога:
Система должна поддерживать до 100 000 SKU (товарных позиций) без снижения скорости фильтрации, поиска и загрузки карточек товаров.
🏥 3. Личный кабинет для записи к врачу на интернет-портале медицинской клиники

Соответствие законодательству (ФЗ-152):
Все персональные данные пациента (ФИО, СНИЛС, история обращений) должны обрабатываться только на серверах, расположенных на территории РФ, с шифрованием данных как при передаче, так и при хранении.
Надёжность записи:
Система должна гарантировать отсутствие двойной записи к одному врачу на одно время даже при высокой нагрузке (например, при открытии записи на популярного специалиста).
Время реакции:
После нажатия кнопки «Записаться» пользователь должен получать подтверждение или ошибку в течение 1 секунды, чтобы избежать повторных кликов и дублирования заявок.
💡 Общий вывод:
Нефункциональные требования критически важны для успеха системы — они определяют, насколько решение будет:

безопасным,
удобным,
надёжным,
соответствующим законам.
В RPA и IT-проектах их нельзя игнорировать, даже если функциональность реализована идеально.

36. Дайте определение понятию «бизнес-процесс»

🔹 Определение:
Бизнес-процесс — это совокупность взаимосвязанных действий (операций), выполняемых в определённой последовательности, направленных на преобразование входов в выходы, имеющие ценность для клиента или организации.

🔹 Ключевые признаки бизнес-процесса:
Направлен на результат — создаёт продукт, услугу или информацию, полезную для внутреннего или внешнего клиента.
Имеет чёткие границы — определены начало (триггер) и конец (результат).
Включает последовательность шагов — действия могут выполнять люди, системы или их комбинация.
Измерим — можно оценить по времени, стоимости, качеству, объёму.
Повторяем — осуществляется регулярно, а не разово.
🔹 Пример:
Процесс «Обработка заказа клиента»:
Вход → Заказ от клиента
Шаги → Проверка наличия → Формирование счёта → Оплата → Отгрузка
Выход → Доставленный товар + удовлетворённый клиент

🔹 В контексте RPA (курс «Атом.РИТА»):
Бизнес-процесс — это объект автоматизации.

Для роботизации он должен быть:

правилозависимым,
повторяющимся,
достаточно стабильным,
выполняемым в цифровой среде.
Источник: Лекции 1, 6

💡 Итог:
Бизнес-процесс — это не просто «работа сотрудника», а структурированная цепочка действий, создающая ценность. Именно такие процессы анализируются, моделируются и при необходимости автоматизируются с помощью RPA.

37. Перечислите участников бизнес процесса

Участники бизнес-процесса (также называемые ролями или акторами) — это физические лица, организации, системы или подразделения, которые выполняют действия в процессе или взаимодействуют с ним.

Согласно материалам курса «Роботизация бизнес-процессов (RPA) на платформе Атом.РИТА» (Лекции 2, 6–8) и общепринятым практикам (BPMN, BABOK), участники делятся на следующие категории:

🔹 1. Исполнители (Performers)
Лица или системы, непосредственно выполняющие шаги процесса:

Сотрудник (например, бухгалтер, менеджер по продажам),
Подразделение (например, отдел кадров, склад),
Информационная система (например, 1С, SAP, CRM),
Программный робот (RPA-бот) — цифровой сотрудник,
Внешняя организация (например, банк, поставщик, почтовый сервис).
Пример: В процессе «Оформление заказа» исполнителями могут быть: менеджер (принимает заказ), ERP-система (резервирует товар), робот (отправляет подтверждение по email).

🔹 2. Заказчик / Получатель результата (Customer)
Внутренний клиент — другой отдел компании (например, финансы получают данные от логистики),
Внешний клиент — конечный потребитель услуги или товара.
Он не участвует в выполнении, но получает ценность от результата процесса.

🔹 3. Владелец процесса (Process Owner)
Ответственное лицо за эффективность, качество и результат процесса.
Утверждает изменения, контролирует KPI.
Часто — руководитель подразделения.
Источник: Лекция 2

🔹 4. Заинтересованные лица (Стейкхолдеры)

ЛПР (лицо, принимающее решение о запуске/финансировании),
Администратор ИБ — контролирует соответствие политикам безопасности,
Владелец ИТ-ресурса — отвечает за систему, с которой взаимодействует процесс,
Конечные пользователи — те, чья работа меняется из-за процесса.
Источник: Лекция 2, Лекция 11

🔹 5. Регуляторы (при необходимости)
Государственные органы,
Аудиторы,
Нормативные требования (например, ФНС, Росздравнадзор).
💡 В контексте моделирования (BPMN):
Участники отображаются как:

Пулы (Pools) — основные стороны (клиент, компания),
Дорожки (Lanes) — роли внутри пула (бухгалтер, робот, 1С).
✅ Итоговый перечень участников бизнес-процесса:
Исполнители (люди, системы, роботы)
Заказчик / Получатель результата
Владелец процесса
ЛПР
Владелец ИТ-ресурса
Администратор информационной безопасности
Конечные пользователи
Внешние организации (поставщики, банки и др.)
Регуляторы (при наличии)

38. Перечислите hard skills и soft skills для профессии «бизнес-аналитик»

🔹 Hard Skills (технические/профессиональные навыки)
Сбор и анализ требований
— Интервью, опросы, наблюдение, workshops.
Моделирование бизнес-процессов
— Владение нотациями: BPMN, IDEF0, eEPC, UML Activity Diagram.
Документирование требований
— Написание User Story, Use Cases, ТЗ, ОПР (Описание программного робота).
Знание методологий разработки
— Waterfall, Agile (Scrum, Kanban), SDLC.
Приоритизация требований
— Применение методов: MoSCoW, RICE, Kano, Wiegers.
Управление требованиями
— Трассировка (матрица трассировки), управление изменениями, версионность.
Основы RPA и автоматизации
— Понимание возможностей/ограничений роботов, работа с платформами (например, Атом.РИТА).
Работа с инструментами
— Jira, Confluence, YouGile, Draw.io, Bizagi, Microsoft Visio, Excel (сводные таблицы, формулы).
Базовое понимание ИТ

— API, базы данных, UI/UX, принципы интеграции систем.
Знание стандартов
— BABOK, ГОСТ 34, ФЗ-152 (для РФ), PCI DSS (при работе с платежами).
🔹 Soft Skills (личностные/коммуникативные навыки)
Коммуникация
— Умение слушать, задавать правильные вопросы, вести переговоры.
Аналитическое и системное мышление
— Декомпозиция сложного, выявление корневых причин, логика.
Управление ожиданиями
— Настройка реалистичных целей у заказчика и команды.
Работа в команде
— Взаимодействие с разработчиками, тестировщиками, владельцами процессов.
Управление временем и приоритетами
— Способность фокусироваться на задачах с высокой ценностью.
Стрессоустойчивость
— Работа в условиях неопределённости, сжатых сроков, конфликтующих требований.
Эмпатия
— Понимание потребностей и болей пользователя/заказчика.
Презентационные навыки
— Умение ясно и убедительно доносить идеи (демо, презентации, согласования).
Критическое мышление
— Способность отличать «желаемое» от «необходимого», избегать предвзятости.
Гибкость и обучаемость
— Быстрое освоение новых предметных областей и технологий.
💡 В контексте RPA (курс Атом.РИТА):
Бизнес-аналитик в RPA-проектах особенно должен:

чётко описывать шаги процесса (для ОПР),
понимать ограничения UI-автоматизации,
уметь рассчитывать эффект (в ПРМ),
эффективно взаимодействовать с владельцами ИТ-ресурсов и службой ИБ.
Как говорится в лекции 2:
«Аналитик — связующее звено между Клиентом и Разработчиком».

✅ Итог
Успешный бизнес-аналитик — это гибрид технического специалиста и коммуникатора. Без hard skills он не сможет структурировать требования, без soft skills — не получит их вообще.

39. Перечислите этапы программной роботизации

🔹 Этапы жизненного цикла программной роботизации (создания ПР — программного робота):
1. Инициация разработки ПР (Offer-card)
Формулировка идеи автоматизации.
Предварительная оценка эффекта (в ПРМ и рублях).
Получение одобрения от ЛПР на запуск аналитики.
Источник: Лекция 11, слайд 3

2. Аналитические работы и планирование
Предпроектное обследование.
Моделирование процесса «as-is» и «to-be».
Сбор требований, идентификация стейкхолдеров.
Подготовка Описания программного робота (ОПР).
Согласование ОПР с владельцем процесса и владельцем ИТ-ресурса.
Источник: Лекции 2, 6–8, 11

3. Процесс создания ПР (разработка + тестирование)
Разработка робота на платформе (например, Атом.РИТА).
Подготовка тестовых данных и сценариев.
Внутреннее тестирование разработчиком.
Совместное тестирование с заказчиком в тестовой среде.
Корректировка по результатам тестирования.
Источник: Лекция 11, слайд 7

4. Внедрение ПР
Получение листов исполнения (ЛИ) от владельцев ИТ-ресурсов.
Настройка доступов в продуктивной среде.
Загрузка робота в продуктив.
Подготовка инструкции пользователя.
Обучение конечных пользователей (при необходимости).
Источник: Лекция 11, слайды 10–12

5. Эксплуатация ПР
Робот работает в штатном режиме.
Мониторинг выполнения (логи, уведомления об ошибках).
Обработка инцидентов Службой поддержки ПР (СППР).
Регулярная проверка корректности работы.
Источник: Лекция 11, слайд 13

6. Вывод ПР из эксплуатации
Прекращение использования робота (процесс упразднён, заменён другим решением и т.д.).
Архивирование кода и документации.
Отзыв доступов.
Формальное закрытие проекта.
Источник: Лекция 11, слайд 14

💡 Дополнительно:
Этапы соответствуют SDLC (Software Development Life Cycle), адаптированному под RPA.
Ключевые артефакты: Offer-card → ОПР → ЛИ → Инструкция → Логи эксплуатации.
Участники: аналитик, разработчик, владелец процесса, владелец ИТ-ресурса, СППР, ЛПР.
✅ Итог
Жизненный цикл программной роботизации — это структурированный процесс, обеспечивающий:

целесообразность автоматизации,
качество реализации,
безопасность внедрения,
устойчивую эксплуатацию.
Без соблюдения этапов велик риск создания неработающего или неподдерживаемого робота.

40. Перечислите базовые объекты нотации BPMN

🔹 Базовые объекты нотации BPMN делятся на четыре основные категории:

1. События (Events)
Обозначают точку начала, промежуточного действия или завершения в процессе.
Изображаются окружностями.

Start Event — начальное событие (триггер запуска процесса).
Intermediate Event — промежуточное событие (ожидание, сообщение, таймер и т.д.).
End Event — конечное событие (успешное завершение, ошибка и т.п.).
Пример: «Получено письмо со счётом» → Start Event.

2. Действия / Задачи (Activities / Tasks)
Представляют работу, выполняемую в процессе.
Изображаются скруглёнными прямоугольниками.

Task — атомарное действие (например, «Сохранить файл»).
Sub-Process — подпроцесс (группа задач, которую можно свернуть/развернуть).
Call Activity — вызов глобального повторно используемого процесса.
Пример: «Извлечь данные из PDF» → Task.

3. Шлюзы (Gateways)
Управляют ветвлением и слиянием потоков на основе условий.
Изображаются ромбами.

Exclusive Gateway (XOR) — выбор одного пути («если… то… иначе…»).
Parallel Gateway (AND) — одновременный запуск нескольких путей.
Inclusive Gateway (OR) — выбор одного или нескольких путей.
Event-Based Gateway — выбор пути по наступившему событию.
Пример: «Есть ли подпись в документе?» → Exclusive Gateway.

4. Потоки (Flow Objects)
Соединяют элементы процесса.

Sequence Flow — сплошная стрелка: порядок выполнения шагов внутри одного пула.
Message Flow — пунктирная стрелка с кружком/стрелкой: обмен сообщениями между пулами (разными участниками).
Association — пунктирная линия без стрелки: связь с артефактами (например, данными или аннотацией).

🔹 Дополнительные элементы (артефакты)
Хотя не являются «базовыми объектами управления», часто используются:

Data Object — данные, используемые или создаваемые в задаче.
Text Annotation — пояснительный текст.
Group — визуальная группировка элементов (без влияния на логику).
🔹 Участники (Swimlanes)
Pool (Пул) — представляет участника процесса (организацию, систему, роль).
Lane (Дорожка) — разделяет обязанности внутри пула (например, «Бухгалтер», «Программный робот»).
Источник: Лекция 6 — моделирование в BPMN

✅ Итог
Базовые объекты BPMN:

События (Events)
Действия / Задачи (Activities)
Шлюзы (Gateways)
Потоки (Sequence Flow, Message Flow)
Эти элементы позволяют точно и наглядно описать логику любого бизнес-процесса, включая процессы, подлежащие роботизации.

41. Подробно расскажите про назначение каждого из объектов:

· Event – Событие;

· Activity – Действия;

· Gateway – Шлюзы или Развилки;

· Flow – Поток.

· Date – Данные;

· Artefact – Артефакты;

· Swimline – «плавательные дорожки»;

· Pool (Пул) — набор.

🔹 1. Event — Событие
Назначение: Обозначает точку, в которой что-то происходит — запуск, промежуточное действие или завершение процесса.
Форма: Окружность.
Типы:

Start Event — триггер начала процесса (например, «Получено письмо со счётом»).
Intermediate Event — событие в середине процесса (ожидание таймера, получение сообщения, ошибка).
End Event — завершение процесса (успешное, с ошибкой, отмена).
💡 В RPA: Start Event часто соответствует триггеру робота (новое письмо, файл в папке).

🔹 2. Activity — Действие (Задача)

Назначение: Представляет работу, выполняемую человеком, системой или роботом.
Форма: Скруглённый прямоугольник.
Типы:

Task — атомарное действие («Сохранить PDF», «Ввести данные в 1С»).
Sub-Process — группа задач, которую можно свернуть для упрощения диаграммы.
Call Activity — вызов глобального, переиспользуемого процесса.
💡 В RPA: Каждый Task может стать шагом в Описании программного робота (ОПР).

🔹 3. Gateway — Шлюз (Развилка)
Назначение: Управляет ветвлением и слиянием потоков выполнения на основе логических условий или параллелизма.
Форма: Ромб.
Основные типы:

Exclusive (XOR) — выбор одного пути («Если сумма > 1 млн → на согласование, иначе — оплатить»).
Parallel (AND) — одновременный запуск всех исходящих потоков («Отправить уведомление + обновить базу»).
Inclusive (OR) — выбор одного или нескольких путей.
Event-Based — выбор пути по наступившему событию («Ждать: либо подтверждение, либо отмену»).
💡 Важно: Шлюзы не выполняют работу — они только направляют поток.

🔹 4. Flow — Поток
Назначение: Определяет порядок выполнения элементов процесса.
Типы:

Sequence Flow (сплошная стрелка) — последовательность внутри одного участника (пула/дорожки).
Message Flow (пунктирная стрелка с кружком/треугольником) — обмен сообщениями между разными пулами (например, клиент → компания).
Association (пунктир без стрелки) — связь с артефактами (данными, аннотациями).
💡 Правило: Sequence Flow не может пересекать границы пулов — для межпульного взаимодействия используется Message Flow.

🔹 5. Data — Данные (Data Object)
Назначение: Отображает входные или выходные данные, используемые/создаваемые задачей.
Форма: Прямоугольник с загнутым углом (как лист бумаги).
Примеры:

Вход: «PDF-счёт»,
Выход: «Заполненная карточка в ERP».
💡 В RPA: Помогает явно указать, какие файлы или структуры данных обрабатывает робот.

🔹 6. Artifact — Артефакт

Назначение: Элементы, не влияющие на логику выполнения, но добавляющие пояснения или контекст.
Типы:

Data Object (см. выше),
Text Annotation — текстовое пояснение («Проверка подписи обязательна»),
Group — визуальная рамка для группировки элементов (без влияния на поток).
💡 Использование: Улучшает читаемость модели для заказчика.

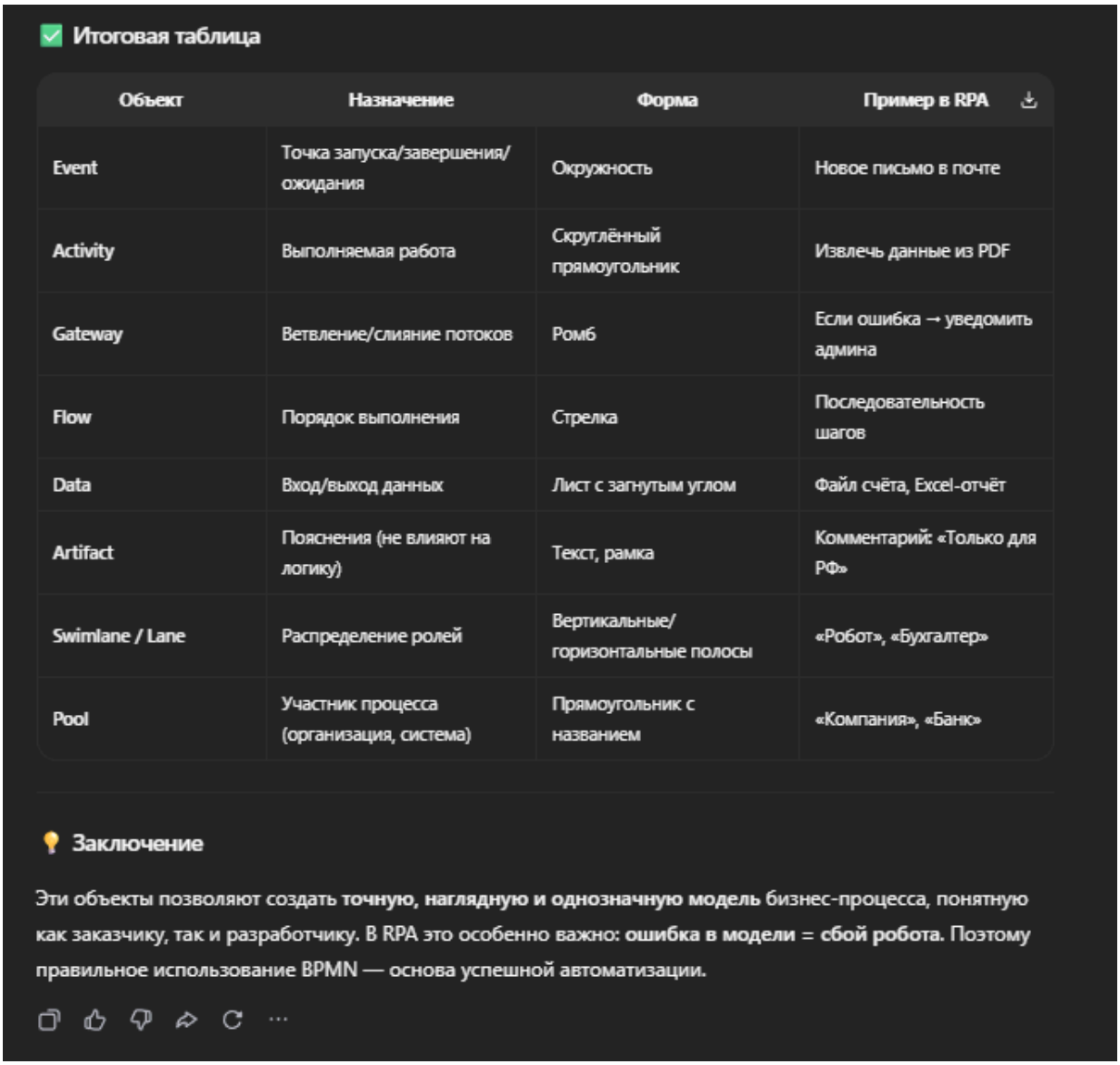
🔹 7. Swimlane — «Плавательные дорожки»
Назначение: Организуют диаграмму по ролям или подразделениям, показывая, кто что делает.
Типы:

Pool (Пул) — представляет основного участника процесса (например, «Клиент», «Компания», «Банк»). Может содержать дорожки.
Lane (Дорожка) — подразделение внутри пула (например, «Бухгалтерия», «Программный робот», «ERP-система»).
💡 В RPA: Чётко разделяет действия человека и робота — критично для согласования ОПР.

🔹 8. Pool (Пул) — Набор / Участник
Назначение: Верхнеуровневый контейнер, представляющий независимую сторону в процессе.
Особенности:

Процесс начинается и заканчивается внутри одного пула.
Взаимодействие между пулами — только через Message Flow.
Может быть раскрытым (содержит детали) или свёрнутым (чёрный ящик).
💡 Пример:

Пул «Клиент» → отправляет заказ.
Пул «Компания» → обрабатывает заказ (внутри: дорожки «Менеджер», «Робот», «1С»).



✅ Итоговая таблица

Объект	Назначение	Форма	Пример в RPA
Event	Точка запуска/завершения/ожидания	Окружность	Новое письмо в почте
Activity	Выполняемая работа	Скруглённый прямоугольник	Извлечь данные из PDF
Gateway	Ветвление/слияние потоков	Ромб	Если ошибка → уведомить админа
Flow	Порядок выполнения	Стрелка	Последовательность шагов
Data	Вход/выход данных	Лист с загнутым углом	Файл счёта, Excel-отчёт
Artifact	Пояснения (не влияют на логику)	Текст, рамка	Комментарий: «Только для РФ»
Swimlane / Lane	Распределение ролей	Вертикальные/горизонтальные полосы	«Робот», «Бухгалтер»
Pool	Участник процесса (организация, система)	Прямоугольник с названием	«Компания», «Банк»

💡 Заключение

Эти объекты позволяют создать **точную, наглядную и однозначную модель** бизнес-процесса, понятную как заказчику, так и разработчику. В RPA это особенно важно: **ошибка в модели = сбой робота**. Поэтому правильное использование BPMN — основа успешной автоматизации.

42. Подробно расскажите для чего нужен объект «подпроцесс»

🔹 Определение

Подпроцесс (Sub-Process) — это составной элемент диаграммы BPMN, который представляет группу связанных задач или шагов, объединённых общей логикой и выделенных в отдельный блок внутри основного процесса.

Формально подпроцесс — это разновидность Activity (действия), но в отличие от простой задачи (Task), он содержит внутреннюю структуру.

🔹 Основные цели и назначение подпроцесса

1. Упрощение и повышение читаемости диаграммы

Когда процесс содержит десятки шагов, его сложно воспринимать целиком.
Подпроцесс позволяет свернуть детали и показать только высокоуровневую логику.
При необходимости можно развернуть подпроцесс и увидеть внутренние шаги.

💡 Пример:

Вместо 10 мелких задач «Проверить подпись → Распознать ИНН → Сверить с контрагентом…» — один блок «Верификация счёта».

2. Логическая группировка операций
Подпроцесс объединяет шаги, имеющие общую цель или выполняемые в рамках одной функции.
Это помогает структурировать процесс по смыслу, а не просто по последовательности.
💡 Примеры групп:

«Подготовка данных»,
«Согласование документа»,
«Обработка ошибок».
3. Повторное использование (через Call Activity)
Если один и тот же набор действий встречается в нескольких местах (например, «Отправка уведомления»), его можно вынести в глобальный подпроцесс.
Затем вызывать его через Call Activity — как функцию в программировании.
⚠️ Обычный (Embedded) Sub-Process не переиспользуется — только Call Activity ссылается на глобальный подпроцесс.

4. Изоляция сложной логики
Внутри подпроцесса можно использовать:
собственные шлюзы (Gateway),
события (Event) (например, таймер или ошибка),
параллельные потоки.
Это не «засоряет» основной процесс и делает модель управляемой.
💡 Пример:
Подпроцесс «Обработка исключения» может содержать ветвление:
— если ошибка критична → уведомить админа,
— если нет → повторить попытку.

5. Управление границами ответственности
Подпроцесс может быть назначен отдельной роли или системе (например, дорожке «Программный робот»).
Это упрощает согласование и распределение обязанностей.
6. Поддержка многоуровневого моделирования
Возможна иерархия подпроцессов: подпроцесс внутри подпроцесса.
Это позволяет строить модели любого уровня детализации — от стратегического до операционного.

🔹 Типы подпроцессов в BPMN

Тип	Описание
Embedded (встроенный) Sub-Process	Часть основного процесса, не существует отдельно. Может быть свёрнут/развёрнут.
Call Activity (вызов подпроцесса)	Ссылка на глобальный (реиспользуемый) подпроцесс, определённый отдельно.
Transactional Sub-Process	Группа действий, которые должны завершиться все успешно или все откатиться (как транзакция в БД).

🔹 Пример из RPA-проекта

Процесс: Автоматическая обработка входящих счетов
Подпроцесс: «Распознавание и валидация данных»
Содержимое подпроцесса:

1. Сохранить PDF во временную папку
2. Запустить OCR
3. Извлечь поля: номер, дата, сумма, ИНН
4. Проверить формат ИНН
5. Сверить сумму с пороговым значением

В основной диаграмме этот блок отображается как один шаг — «Распознать и проверить данные», что делает модель чище и понятнее для владельца процесса.

🔹 Визуальное представление

- Подпроцесс выглядит как обычная задача, но с плюсом (+) в левом нижнем углу, когда свёрнут.
- При разворачивании внутри него отображаются все элементы BPMN: события, задачи, шлюзы.

✅ Итог

Подпроцесс нужен для:

- упрощения сложных диаграмм,
- логической группировки шагов,
- повторного использования логики,
- изоляции обработки исключений,
- многоуровневого моделирования.

В проектах RPA подпроцессы особенно полезны для выделения стандартных блоков автоматизации (например, «работа с почтой», «обработка Excel»), что ускоряет разработку и повышает надёжность ОПР.

43. Рассказать про интерфейс студии разработки Атом.РИТА.

Ответ основан на официальной документации и типичной структуре RPA-платформ, а также на логике, заложенной в курсе «Роботизация бизнес-процессов (RPA) на платформе Атом.РИТА» (Лекции 10–16).

⚠️ Примечание: точный внешний вид интерфейса может отличаться в зависимости от версии платформы, но основные компоненты остаются стандартными для большинства RPA-студий.

🔹 Общее назначение Студии разработки Атом.РИТА
Студия разработки — это среда, в которой разработчик программных роботов (ПР) создаёт, тестирует и отлаживает автоматизацию на основе Описания программного робота (ОПР).

🔹 Основные элементы интерфейса Студии Атом.РИТА
1. Панель проекта / Дерево активностей (Activity Tree)
Отображает структуру робота в виде иерархического дерева.
Каждый шаг (действие) — это активность (например, «Открыть браузер», «Нажать кнопку», «Прочитать ячейку Excel»).
Поддерживает вложенные блоки: последовательности (Sequence), циклы (While, For Each), обработка ошибок (Try Catch).
💡 Аналог: дерево кода в Visual Studio или PyCharm.

2. Холст проектирования (Design Canvas)
Визуальная область, где разработчик перетаскивает и соединяет активности.
Представляет логику робота в виде блок-схемы.
Поддерживает группировку, комментарии, цветовую маркировку для удобства чтения.
3. Палитра активностей (Activities Panel / Toolbox)
Список готовых строительных блоков для робота.
Группируется по категориям:
UI Automation (работа с окнами, клики, ввод текста),
Excel / Word / PDF (работа с офисными файлами),
Email (Outlook, SMTP),
Database (SQL-запросы),
Flow Control (циклы, условия, исключения),
System (работа с файлами, папками, процессами).
💡 В Атом.РИТА активности часто называются «действиями» или «шагами».

4. Панель свойств (Properties Panel)
Отображается при выделении конкретной активности.
Позволяет настроить параметры действия:
целевой элемент (селектор, XPath),
входные/выходные переменные,
таймауты,

условия повтора,
обработка ошибок.
Пример: для действия «Нажать кнопку» указывается селектор //button[@id='submit'].

5. Панель переменных и аргументов (Variables & Arguments)
Здесь объявляются:
Переменные (локальные данные: строки, числа, таблицы),
Аргументы (входные/выходные параметры робота).
Поддерживается типизация (String, Int32, DataTable и др.).
6. Панель вывода и отладки (Output / Debug Panel)
Отображает:
лог выполнения,
значения переменных в реальном времени,
ошибки и предупреждения.
Используется при пошаговом выполнении (Debug).
7. Панель управления запуском (Run / Debug Toolbar)
Кнопки:
Run — запуск всего робота,
Debug — пошаговый запуск,
Stop — остановка,
Breakpoint — установка точек останова.
8. Менеджер ресурсов / Управление зависимостями
Позволяет подключать:
внешние библиотеки,
файлы конфигурации,
учётные данные (через безопасное хранилище),
сертификаты.
9. Интеграция с порталом Атом.РИТА
Возможность:
загрузить ОПР напрямую из портала,
отправить робота на тестирование,
опубликовать в продуктивную среду.
Источник: Лекция 11 — жизненный цикл ПР

🔹 Особенности Атом.РИТА (в сравнении с другими RPA-платформами)
Ориентация на российские корпоративные стандарты (ГОСТ, ФЗ-152).
Тесная интеграция с отечественными системами: 1С, СЭД, госпорталы.
Поддержка работы с CAPTCHA и двухфакторной аутентификацией через внешние сервисы (2Captcha и др.).
Встроенные средства логирования и аудита — критично для регуляторных требований.
💡 Пример рабочего процесса в Студии:
Разработчик открывает ОПР из портала.
Перетаскивает активности с палитры на холст.
Настраивает селекторы и переменные в панели свойств.
Запускает робота в режиме отладки.
Проверяет лог и корректирует ошибки.
Публикует робота для тестирования.
✅ Итог

Интерфейс Студии разработки Атом.РИТА построен по принципу визуального программирования:
— разработчик не пишет код, а собирает робота из готовых блоков,
— всё управление — через настройку свойств и логики потока,
— акцент сделан на безопасность, соответствие требованиям и интеграцию с российскими ИТ-ландшафтами.

Это позволяет даже неопытным разработчикам быстро создавать надёжных программных роботов на основе чётко описанного ОПР.

44. Переменные студии разработки, способы добавления, правила наименования. Типы данных. Логические выражения и операторы

🔹 1. Что такое переменные в Студии разработки?
Переменная — это именованная область памяти, используемая для хранения данных, которые могут изменяться в процессе выполнения робота.
Переменные позволяют:

передавать данные между действиями,
сохранять промежуточные результаты,
управлять логикой выполнения (условия, циклы).

🔹 2. Способы добавления переменных в Атом.РИТА
В большинстве RPA-студий (включая Атом.РИТА) переменные создаются через панель переменных:

✅ Способ 1: Через панель Variables
Открыть вкладку «Variables» (обычно внизу или справа).
Нажать «+» или «Add Variable».
Указать:
Имя (Name),
Тип данных (Type),
Значение по умолчанию (Default value, опционально),
Область видимости (Scope — например, Sequence, Flowchart, весь проект).
✅ Способ 2: Автоматическое создание
При указании выходного параметра действия (например, Output → Text) можно ввести новое имя переменной — студия предложит её создать автоматически.

🔹 3. Правила наименования переменных
Хотя точные правила зависят от платформы, в Атом.РИТА рекомендуются следующие практики (аналогично C#, Java):

Имя должно начинаться с буквы или символа подчёркивания _.
Допустимы буквы, цифры, _.
Нельзя использовать пробелы и специальные символы (@, #, $, - и т.д.).
Регистр имеет значение: counter ≠ Counter.
Используется camelCase или PascalCase:
invoiceNumber — для локальных переменных,
InvoiceTotal — для значимых данных.

Имена должны быть самодокументируемыми:
❌ x1, temp → ✅ emailAddress, totalAmount.
⚠️ Зарезервированные слова (например, if, true, string) использовать нельзя.

🔹 4. Основные типы данных в Атом.РИТА

Тип	Описание	Пример
String	Текстовая строка	`"Привет"`, `"invoice_001.pdf"`
Int32 / Integer	Целое число	`42`, `-5`, `1000`
Double / Decimal	Число с плавающей точкой	`123.45`, `0.99`
Boolean	Логическое значение	`True`, `False`
DateTime	Дата и время	`25.01.2026 10:30:00`
DataTable	Таблица данных (аналог Excel/SQL-таблицы)	Результат запроса к БД
Array	Массив элементов одного типа	`{"Москва", "СПб", "ЕКБ"}`
GenericValue	Универсальный тип (может хранить любой тип)	Используется при неопределённости

> 💡 В Атом.РИТА также поддерживаются кастомные объекты и словари (Dictionary) в продвинутых сценариях.

🔹 5. Логические выражения и операторы

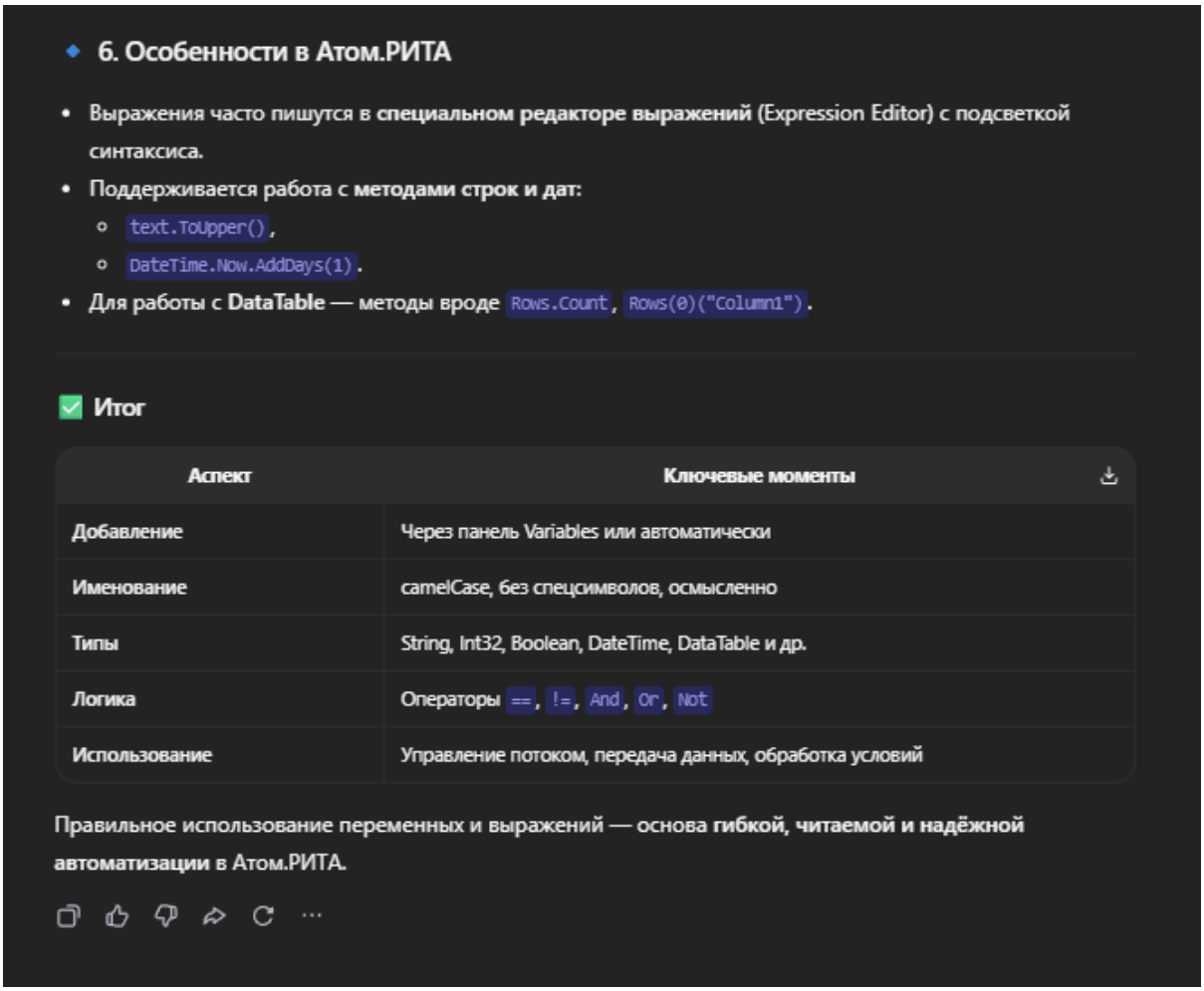
Логические выражения используются в условиях (`If`, `While`) и возвращают **Boolean** (`True` / `False`).

✅ Операторы сравнения

Оператор	Значение	Пример
`==`	Равно	`counter == 10`
`!=`	Не равно	`status != "Готово"`
`>` / `<`	Больше / меньше	`amount > 1000`
`>=` / `<=`	Больше/равно, меньше/равно	`daysLeft <= 0`

✅ Логические операторы

Оператор	Значение	Пример
`And`	И	`(age >= 18) And (hasLicense == True)`
`Or`	Или	`(errorType == "A") Or (errorType == "B")`
`Not`	Не	`Not isProcessed`



🔹 6. Особенности в Атом.РИТА

- Выражения часто пишутся в **специальном редакторе выражений** (Expression Editor) с подсветкой синтаксиса.
- Поддерживается работа с **методами строк и дат**:
  - `text.ToUpper()`,
  - `DateTime.Now.AddDays(1)`.
- Для работы с **DataTable** — методы вроде `Rows.Count`, `Rows(0)("Column1")`.

✅ Итог

Аспект	Ключевые моменты
Добавление	Через панель Variables или автоматически
Именование	camelCase, без спецсимволов, осмысленно
Типы	String, Int32, Boolean, DateTime, DataTable и др.
Логика	Операторы `==`, `!=`, `And`, `Or`, `Not`
Использование	Управление потоком, передача данных, обработка условий

Правильное использование переменных и выражений — основа **гибкой, читаемой и надёжной автоматизации** в Атом.РИТА.

45. Понятие активность в студии разработки платформы Атом.РИТА. Виды активностей. Действия с активностью. Настройка активности

46. Условие и циклы в студии разработки платформы Атом.РИТА. Привести примеры использования. Основные параметры заполнения активности «Цикл для каждого». Основные параметры заполнения активности «Условие True/false».

47. Что такое логирование? Возможности отладки и контроля робота. Добавление точки останова.

🔹 1. Что такое логирование?

Логирование — это процесс записи событий, действий и состояний, происходящих во время выполнения программного робота.

✅ Цели логирования:

Отслеживать последовательность выполнения шагов.

Фиксировать значения переменных в ключевые моменты.
Диагностировать ошибки и исключения.
Обеспечивать аудит и прозрачность (особенно важно в финансах, госсекторе).
Подтверждать факт выполнения операций (для compliance).
💡 Пример записи в лог:
123
В Атом.РИТА логирование реализуется через специальную активность (например, «Запись в лог» или Log Message), где можно указать:

Уровень: Info, Debug, Warning, Error,
Сообщение (включая значения переменных),
Формат вывода (консоль, файл, централизованная система мониторинга).
🔹 2. Возможности отладки и контроля робота
Студия разработки Атом.РИТА предоставляет инструменты для пошагового анализа и проверки логики:

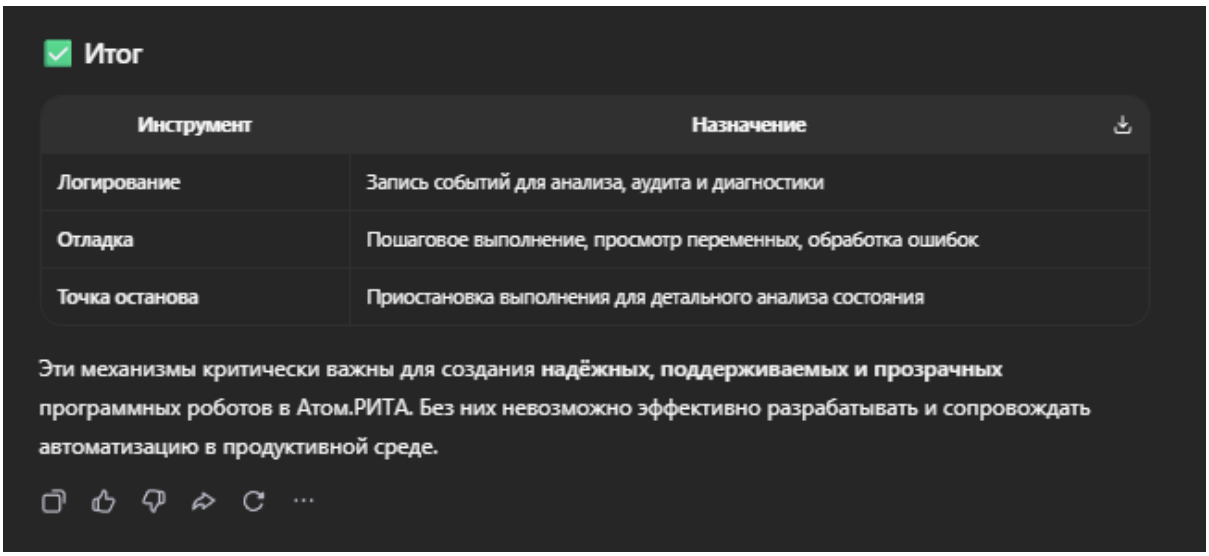
✅ Основные возможности:
Пошаговое выполнение (Step Into / Step Over)
— Робот выполняет одно действие за раз, позволяя наблюдать изменения состояния.
Просмотр значений переменных в реальном времени
— В панели Variables или при наведении курсора отображаются текущие значения.
Лог-панель (Output / Log Panel)
— Отображает все сообщения, ошибки, предупреждения во время выполнения.
Обработка исключений (Try Catch)
— Позволяет «ловить» ошибки и выполнять альтернативные действия (например, отправить уведомление).
Мониторинг выполнения в Orchestrator (Портал Атом.РИТА)
— После публикации робота его запуски, статусы и логи доступны в веб-интерфейсе для службы поддержки.
🔹 3. Добавление точки останова (Breakpoint)
Точка останова — это маркер, который приостанавливает выполнение робота на определённом шаге для анализа состояния.

✅ Как добавить точку останова в Атом.РИТА:
Откройте проект в Студии разработки.
Найдите нужную активность на холсте (например, «Прочитать ячейку Excel»).
Щёлкните по значку слева от активности (обычно появляется красный кружок)
ИЛИ
ПКМ → «Toggle Breakpoint».
Запустите робота в режиме Debug (не Run!).
✅ Что происходит при достижении точки останова:
Выполнение приостанавливается.
Можно:
осмотреть значения всех переменных,
проверить состояние UI (если робот работает с окнами),
продолжить выполнение пошагово.
⚠️ Точки останова работают только в режиме отладки и игнорируются при обычном запуске.

🔹 Практический пример в RPA-проекте
Сценарий: Робот обрабатывает счета, но иногда пропускает файлы.

Действия разработчика:

Добавляет логирование перед и после чтения файла:
Log("Обрабатываю файл: " + fileName)
Ставит точку останова на шаге «Распознать PDF».
Запускает в Debug-режиме.
Видит, что fileName = "" → ошибка в предыдущем шаге.
Исправляет логику выбора файла.



✅ Итог

Инструмент	Назначение
Логирование	Запись событий для анализа, аудита и диагностики
Отладка	Пошаговое выполнение, просмотр переменных, обработка ошибок
Точка останова	Приостановка выполнения для детального анализа состояния

Эти механизмы критически важны для создания **надёжных, поддерживаемых и прозрачных** программных роботов в Атом.РИТА. Без них невозможно эффективно разрабатывать и сопровождать автоматизацию в продуктивной среде.

Модуль «Управление проектами в RPA»

1 Назовите обязательные характеристики, которым должен соответствовать планируемый для роботизации процесс

🔹 Обязательные характеристики процесса, пригодного для роботизации:
1. Правилозависимость (Rule-based)
Процесс должен выполняться по чётким, формализованным правилам, а не на основе субъективных решений или интуиции.
Пример: «Если сумма счёта > 1 млн → отправить на согласование» — ✅
«Оценить, выглядит ли документ подозрительно» — ❌
2. Повторяемость (Repetitive)
Процесс должен регулярно повторяться: ежедневно, еженедельно, при наступлении события.
Робот эффективен там, где есть объём однотипных операций.
Пример: обработка 50+ счетов в день — ✅
Разовая миграция данных — ❌ (не окупится)

3. Стабильность (Stable)
Логика процесса и интерфейсы используемых систем не должны часто меняться. Частые обновления 1С, смена дизайна веб-сайта или почтового клиента могут ломать робота.
Пример: стандартный процесс сверки выписок — ✅
Работа с пилотным сервисом, который меняется каждую неделю — ❌
4. Цифровая среда выполнения (Digital)
Все шаги процесса должны происходить в цифровых системах (без бумажных носителей, сканов без OCR, устных согласований).
Робот не может:
читать бумажные документы,
звонить по телефону (без интеграции VoIP),
принимать решения на основе эмоций.
Пример: работа с Excel, 1С, Outlook, веб-порталами — ✅
Обработка бумажных анкет — ❌
5. Структурированность данных (Structured Input)
Входные данные должны быть предсказуемыми по формату:
— фиксированные шаблоны документов,
— стандартные поля в формах,
— регулярная структура файлов (Excel, XML, JSON).
Пример: PDF-счета одного шаблона — ✅
Сканы договоров в произвольной форме — ❌ (без ИИ/ML)
💡 Дополнительно (из курса):
Процесс должен приносить измеримый эффект, выраженный в:

ПРМ (полных рабочих месяцах),
рублях (экономия на ФОТ, снижение штрафов).
Как говорится в Лекции 1:
«Автоматизировать нужно не всё подряд, а только то, что приносит измеримую пользу и соответствует критериям роботизации».

✅ Итог
Процесс пригоден к роботизации, если он:

Правилозависим
Повторяющийся
Стабильный
Цифровой
Структурированный
Без этих характеристик автоматизация будет дорогой, хрупкой и неэффективной.

2 Заинтересованные стороны проекта (стейкхолдеры) - это

🔹 Определение:
Заинтересованные стороны (стейкхолдеры) — это физические лица, группы или организации, которые:

оказывают влияние на проект,

испытывают влияние от его результатов,
имеют интерес в успехе или неудаче проекта.
Источник: Лекция 2, курс «Роботизация бизнес-процессов (RPA) на платформе Атом.РИТА»

🔹 Основные категории стейкхолдеров в RPA-проекте:
Лицо, принимающее решение (ЛПР)
— Утверждает запуск проекта, выделяет бюджет.
Владелец процесса
— Отвечает за эффективность и результат бизнес-процесса, подлежащего автоматизации.
Владелец ИТ-ресурса
— Отвечает за систему (1С, Outlook, СЭД и др.), с которой будет взаимодействовать робот.
Функциональный заказчик / Инициатор
— Тот, кто предложил идею автоматизации (часто совпадает с владельцем процесса).
Конечный пользователь
— Сотрудник, чья работа изменяется из-за внедрения робота.
Разработчик ПР (СРПР)
— Создаёт программного робота на основе ОПР.
Служба поддержки ПР (СППР)
— Обеспечивает мониторинг, устранение сбоев, сопровождение в продуктиве.
Администратор информационной безопасности (ИБ)
— Проверяет соответствие робота политикам безопасности (учётные данные, доступы).
Эксперт функционального направления (ЭФН)
— Гарантирует, что логика робота не конфликтует с существующими бизнес-правилами.
Владелец учётной записи программного робота
— Формальный ответственный за «цифрового сотрудника».
🔹 Зачем их идентифицировать?
Чтобы правильно собрать требования.
Чтобы обеспечить согласование на всех этапах.
Чтобы избежать сопротивления и сбоев при внедрении.
Чтобы распределить роли и зоны ответственности.
Источник: Лекции 2, 11

✅ Итог:
Стейкхолдеры — это все, кто так или иначе вовлечён в проект или затронут им.
Успешная роботизация невозможна без их учёта, вовлечения и управления ожиданиями.

3 Сформулируйте цель управления заинтересованными стейкхолдерами

🔹 Цель управления заинтересованными сторонами (стейкхолдерами):
Обеспечить своевременное вовлечение, согласование и поддержку всех ключевых участников проекта для достижения его целей с минимальными рисками, конфликтами и изменениями на поздних этапах.

🔹 Развёрнуто, цель включает:
Идентификацию всех стейкхолдеров на раннем этапе.
Анализ их интересов, влияния и ожиданий (например, через матрицу влияния/интереса).
Выстраивание эффективной коммуникации, соответствующей роли каждого участника.
Управление ожиданиями — чёткое объяснение возможностей, сроков и ограничений.
Получение необходимых согласований (от владельца процесса, ИТ-ресурса, ИБ и др.).
Снижение сопротивления изменениям (особенно со стороны конечных пользователей).
Обеспечение устойчивой поддержки проекта на всех этапах — от инициации до эксплуатации.
Источник: Лекция 2, Лекция 5 — работа со стейкхолдерами и матрица приоритизации

💡 Пример из RPA:
Если не согласовать автоматизацию с владельцем ИТ-ресурса, робот не получит доступ к системе.
Если не вовлечь конечного пользователя, он может sabotировать работу (например, изменять формат файлов).
Управление стейкхолдерами предотвращает такие риски.

✅ Краткая формулировка цели:
Достичь успеха проекта через проактивное взаимодействие с теми, кто может повлиять на него или быть им затронут.

4 Что такое матрица стейкхолдеров?

🔹 Определение:
Матрица стейкхолдеров — это инструмент управления заинтересованными сторонами, который позволяет визуально классифицировать участников проекта по двум ключевым параметрам:

Уровень влияния (насколько они могут повлиять на проект),
Уровень интереса (насколько им важен результат проекта).
Цель матрицы — определить стратегию взаимодействия с каждой группой стейкхолдеров.

Источник: Лекция 5, курс «Роботизация бизнес-процессов (RPA) на платформе Атом.РИТА»

🔹 Структура матрицы (четыре квадранта):

Квадрант	Уровень влияния	Уровень интереса	Стратегия взаимодействия
1. Управлять внимательно	Высокий	Высокий	Активно вовлекать, регулярно согласовывать, информировать детально.
2. Держать в курсе	Низкий	Высокий	Регулярно информировать, учитывать мнение, но не перегружать деталями.
3. Информировать минимально	Низкий	Низкий	Минимальная коммуникация — только ключевые события.
4. Удовлетворять	Высокий	Низкий	Поддерживать лояльность, периодически информировать, не допускать недовольства.

🔹 Примеры стейкхолдеров в RPA-проекте:
Квадрант 1 (Управлять внимательно):
— Владелец процесса,
— ЛПР (лицо, принимающее решение),
— Владелец ИТ-ресурса.
Квадрант 2 (Держать в курсе):
— Конечные пользователи,
— Коллеги из смежных отделов.
Квадрант 3 (Информировать минимально):
— Сотрудники, косвенно затронутые (например, ИТ-поддержка без прямого участия).
Квадрант 4 (Удовлетворять):
— Топ-менеджмент (если не вовлечён напрямую, но может остановить проект),
— Администратор ИБ.
🔹 Зачем нужна матрица?
Избежать перекоммуникации (не тратить время на тех, кому это не нужно).
Не упустить ключевых лиц, чьё несогласие может остановить проект.
Эффективно распределить ресурсы на коммуникацию.
Повысить прозрачность и управляемость проекта.
✅ Итог:
Матрица стейкхолдеров — это практический инструмент приоритизации коммуникаций, помогающий бизнес-аналитику и менеджеру проекта фокусироваться на самом важном и обеспечивать поддержку проекта со стороны всех значимых участников.

5 Какая стратегия взаимодействия актуальна для стейкхолдеров с низкой заинтересованностью и высоким влиянием?

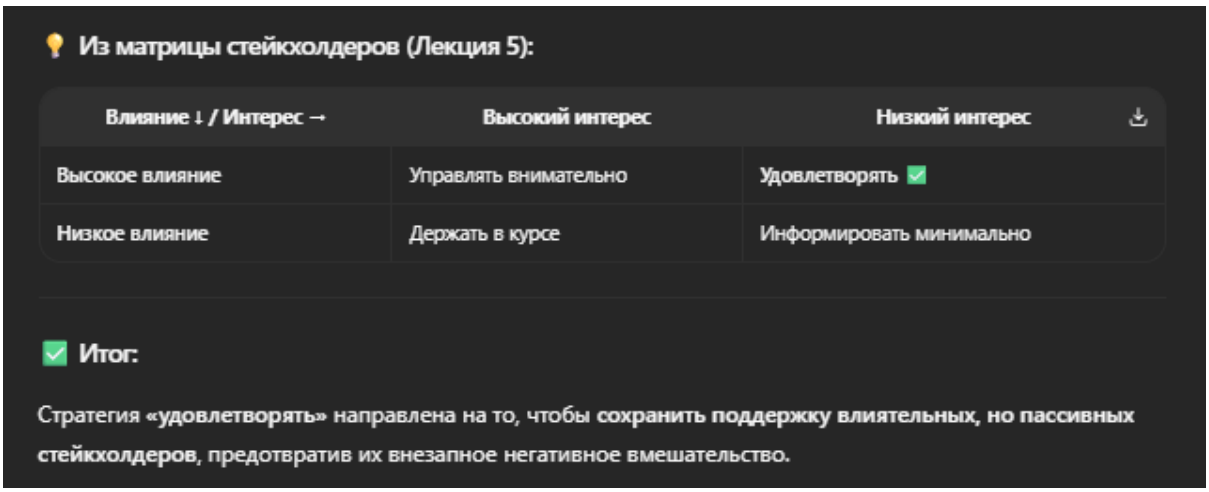
🔹 Ответ:
Для стейкхолдеров с низким уровнем заинтересованности и высоким уровнем влияния применяется стратегия:

«Удовлетворять»

🔹 Пояснение:
Такие стейкхолдеры:

Могут существенно повлиять на проект (например, остановить его, отозвать ресурсы, изменить приоритеты),
Но не проявляют активного интереса к деталям или ходу выполнения.
Если их игнорировать, они могут внезапно вмешаться — например, при появлении жалоб или перераспределении бюджета.

🔹 Рекомендуемые действия:
Регулярно информировать о ключевых результатах (кратко, без технических деталей),
Подчёркивать выгоды и достижения проекта,
Поддерживать позитивное отношение,
Не допускать накопления недовольства.
🔹 Примеры таких стейкхолдеров в RPA-проекте:
Топ-менеджер, утвердивший бюджет, но не участвующий в деталях,
Руководитель ИТ-департамента, выделивший инфраструктуру, но не вовлечённый в процесс,
Администратор информационной безопасности (может заблокировать доступ, если посчитает риски неприемлемыми).



6 Риск в управлении проектами – это

🔹 Определение:
Риск в управлении проектами — это неопределённое событие или условие, которое, если произойдёт, окажет позитивное или негативное влияние на цели проекта (сроки, бюджет, качество, объём работ).

⚠️ Хотя часто под «риском» понимают только угрозы, в профессиональном управлении проектами риск может быть и возможностью (позитивным риском).

🔹 Ключевые характеристики риска:
Неопределённость — событие ещё не произошло и может как случиться, так и не случиться.
Вероятность — степень возможности наступления события (низкая, средняя, высокая).

Влияние (последствия) — масштаб воздействия на проект (время, деньги, репутация и т.д.).

🔹 Примеры рисков в RPA-проекте:

❌ Негативные риски (угрозы):

Обновление интерфейса 1С → робот перестаёт работать.

Отказ владельца ИТ-ресурса предоставить доступ → невозможность автоматизации.

Утечка учётных данных робота → нарушение политики ИБ.

✅ Позитивные риски (возможности):

Появление API у системы → можно заменить UI-автоматизацию на более надёжную интеграцию.

Расширение процесса → робот можно масштабировать на другие подразделения.

🔹 Цель управления рисками:

Выявить потенциальные риски заранее,

Оценить их вероятность и влияние,

Разработать меры реагирования:

Избежать (avoid),

Передать (transfer),

Смягчить (mitigate),

Принять (accept) — для угроз;

Использовать, усилить, делегировать — для возможностей.

Источник: Лекция 5 — управление рисками при приоритизации и планировании

✅ Итог:

Риск — это не обязательно проблема, а фактор неопределённости, который требует проактивного управления.

Эффективное управление рисками повышает шансы на успешную реализацию проекта, особенно в таких чувствительных к изменениям областях, как RPA.

7 Как зависит возможность влияния руководителя проекта на риски с течением времени?
Как зависит ущерб от риска с течением времени при реализации проекта?

🔹 1. Возможность влияния на риски во времени

Чем раньше в жизненном цикле проекта выявлен и обработан риск, тем выше возможность на него повлиять.

На начальных этапах (инициация, планирование):

Руководитель проекта имеет максимальную гибкость:

может изменить требования,

выбрать другую технологию,

перераспределить ресурсы,

отказаться от нереалистичных целей.

Стоимость корректировок — минимальна.

По мере продвижения проекта (разработка, тестирование, внедрение):

Архитектура, логика и интеграции уже зафиксированы.

Любые изменения требуют переделки, что дорого и трудоёмко.

Влияние руководителя снижается экспоненциально.

📉 Графически: кривая «влияние на риск» — убывающая с течением времени.

🔹 2. Ущерб от реализации риска во времени
Чем позже проявляется риск, тем выше его потенциальный ущерб для проекта.

На ранних этапах:
Ошибка в требованиях или оценке — легко исправляется.
Ущерб: потеря нескольких часов аналитика.
На поздних этапах (внедрение, эксплуатация):
Сбой робота в продуктиве → остановка бизнес-процесса.
Нарушение безопасности → штрафы, репутационные потери.
Необходимость полной переделки → срыв сроков, бюджета.
📈 Графически: кривая «ущерб от риска» — возрастающая с течением времени.

🔹 **Иллюстрация (из практики RPA):**

Этап	Риск	Возможность влияния	Ущерб
Анализ	Неверно определён формат PDF-счёта	Высокая — можно уточнить у заказчика	Низкий — коррекция ОПР
Разработка	Выбран нестабильный метод распознавания	Средняя — можно переписать модуль	Средний — задержка на 2 дня
Эксплуатация	Робот теряет данные из-за ошибки	Низкая — процесс уже запущен	Высокий — финансовые потери, недоверие

💡 **Ключевой вывод (из PMBOK и практики RPA):**

> «Раннее выявление рисков — самый эффективный способ управления ими».

Поэтому в RPA-проектах так важны:

- качественное **предпроектное обследование**,
- чёткое **Описание программного робота (ОПР)**,
- **совместное тестирование** до внедрения.

✅ **Итог:**

- **Возможность влияния на риск** ↓ (уменьшается) с течением времени.
- **Ущерб от риска** ↑ (увеличивается) с течением времени.

Это фундаментальный принцип управления проектами и основание для **проактивного подхода к рискам.**

8 Что в данном примере является риском? Вследствие роста курса доллара могут измениться цены на серверное оборудование, что может привести к превышению выделенного бюджета проекта

Риском является событие:

«Рост курса доллара».

🔹 Пояснение:
В управлении проектами риск — это неопределённое событие или условие, которое при наступлении влияет на цели проекта.

В данном случае:

Неопределённое событие: рост курса доллара (пока не произошло, но возможно),
Последствие (влияние): изменение цен на серверное оборудование → превышение бюджета.
Таким образом:

Риск = причина (рост курса),
Ущерб / последствие = эффект (перерасход бюджета).
💡 Это классический пример финансового риска с внешней (макроэкономической) природой.

📌 Дополнительно (по терминологии PMBOK и RPA-практик):
Триггер риска: публикация ЦБ нового курса.
Мера реагирования: фиксация цен у поставщика, закупка авансом, использование оборудования в облаке (альтернатива).
✅ Итог:
В приведённом примере риском является «рост курса доллара», так как это неопределённое событие, способное негативно повлиять на бюджет проекта.

9 Как рассчитывается величина риска?

🔹 **Формула расчёта величины (уровня) риска:**

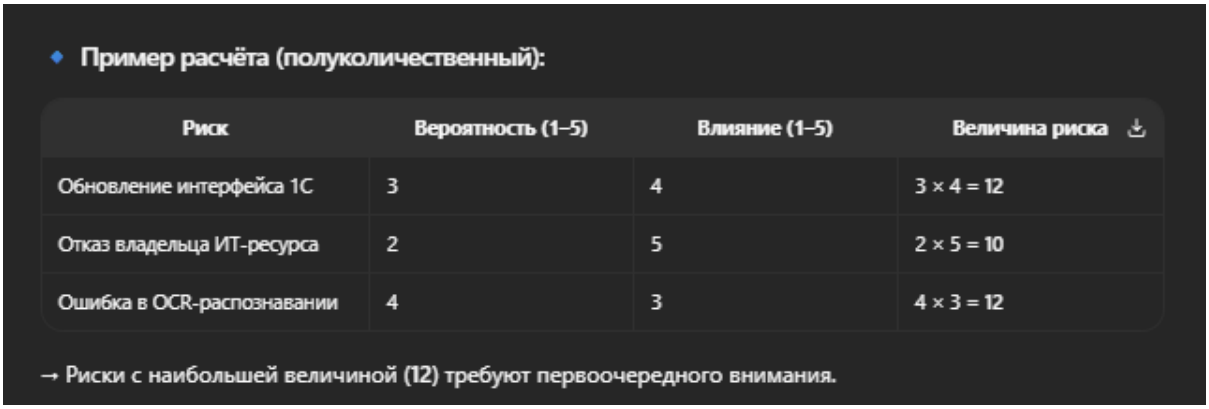
$$\text{Величина риска} = \text{Вероятность наступления} \times \text{Влияние (ущерб)}$$

Это стандартный подход, используемый в управлении проектами (включая RPA-проекты) для количественной или полуколичественной оценки рисков.

🔹 Компоненты формулы:
1. Вероятность (Probability)
— Насколько вероятно, что риск произойдёт.
Может выражаться:

в процентах (например, 30%),
по шкале (например, от 1 до 5: 1 — очень низкая, 5 — очень высокая).
2. Влияние / Ущерб (Impact)
— Насколько сильно риск повлияет на проект, если произойдёт.
Оценивается по ключевым параметрам:

бюджет,
сроки,
качество,
репутация.
Также может быть:
количественным (например, убыток в ₽500 000),
качественным (по шкале от 1 до 5).



🔹 **Пример расчёта (полуколичественный):**

Риск	Вероятность (1–5)	Влияние (1–5)	Величина риска
Обновление интерфейса 1С	3	4	3 × 4 = 12
Отказ владельца ИТ-ресурса	2	5	2 × 5 = 10
Ошибка в OCR-распознавании	4	3	4 × 3 = 12

→ Риски с наибольшей величиной (12) требуют первоочередного внимания.

🔹 Использование в RPA-проектах (Лекция 5):
Помогает приоритизировать риски при планировании.
Основа для разработки планов реагирования:
Высокий риск → меры по снижению вероятности или влияния.
Низкий риск → принятие (monitoring).
💡 Важно:
Если используется качественная шкала, результат — условный показатель (не рубли!).
Для точного финансового анализа применяется EMV (Expected Monetary Value):
EMV
=
Вероятность (%)
×
Финансовый ущерб
EMV=Вероятность (%)×Финансовый ущерб
Пример: 20% × ₽1 000 000 = ₽200 000 — ожидаемые потери.

✅ Итог:
Величина риска = Вероятность × Влияние.
Этот расчёт позволяет объективно сравнивать риски и эффективно распределять ресурсы на их управление.

10 Назовите минимум две стратегии реагирования на риск

✅ Две (и более) стратегии реагирования на риск:
1. Избежать (Avoid)
— Устранить угрозу, изменив план проекта.
Пример в RPA: Отказаться от автоматизации процесса в системе с частыми обновлениями интерфейса — выбрать более стабильную альтернативу.

2. Смягчить (Mitigate)
— Снизить вероятность или влияние риска.
Пример: Добавить в робота механизм повторной попытки и уведомления при сбое подключения к 1С.

3. Передать (Transfer)
— Переложить ответственность за риск на третью сторону.
Пример: Использовать облачный OCR-сервис с SLA вместо собственного распознавания — ответственность за качество ложится на поставщика.

4. Принять (Accept)
— Осознанно признать риск и не предпринимать активных действий (часто для низких рисков).
Пример: Принять, что при редком изменении макета Excel-отчёта робот остановится — и обрабатывать такие случаи вручную.

💡 Для позитивных рисков (возможностей) применяются стратегии: Использовать, Усилить, Делегировать, Принять.

🔹 Источник:
Лекция 5 курса «Роботизация бизнес-процессов (RPA) на платформе Атом.РИТА» — управление рисками при планировании и приоритизации.

✅ Итог:
Минимум две стратегии:

Избежать
Смягчить
(Также допустимы: передать, принять — в зависимости от контекста.)

11 Что такое иерархическая структура работ?

🔹 Определение:
Иерархическая структура работ (ИСР) — это дерево задач проекта, в котором весь объём работ разбит на более мелкие, управляемые компоненты, организованные в иерархию от общего к частному.

На английском — WBS (Work Breakdown Structure).

Источник: Лекция 5, курс «Роботизация бизнес-процессов (RPA) на платформе Атом.РИТА»

🔹 Цель ИСР:
Чётко определить весь объём работ проекта.
Разделить сложный проект на логические, измеримые части.
Обеспечить основу для:
планирования сроков,
оценки ресурсов и бюджета,
назначения ответственных,

контроля выполнения.

🔹 Принципы построения ИСР:

100%-ное правило: ИСР включает весь объём работ проекта и только его (ничего лишнего).

Взаимоисключающие элементы: задачи не дублируют друг друга.

Детализация до уровня пакета работ — такой единицы, которую можно:

оценить по времени и стоимости,

назначить исполнителю,

измерить результат.

🔹 Формат представления:

Древовидная диаграмма,

Нумерованный список,

Таблица.

💡 Важно:

ИСР не показывает последовательность или зависимости между задачами (это — сетевой график или диаграмма Ганта).

ИСР отвечает только на вопрос: «Что нужно сделать?», а не «Когда и в каком порядке?».

✅ Итог:

Иерархическая структура работ (ИСР) — это фундамент планирования проекта, обеспечивающий полноту охвата работ и возможность эффективного управления ими. В RPA-проектах она помогает структурировать процесс от идеи до эксплуатации программного робота.

12 Что такое матрица RACI?

🔹 Определение:

Матрица RACI — это инструмент управления проектами, который чётко распределяет роли и зоны ответственности участников за каждую задачу или этап проекта.

Название RACI происходит от первых букв английских ролей:

Буква	Роль	Описание
R — Responsible (Исполнитель)	Тот, кто непосредственно выполняет работу. Может быть несколько человек.	
A — Accountable (Ответственный)	Тот, кто несёт окончательную ответственность за результат. Принимает решение, утверждает. Обычно один на задачу.	
C — Consulted (Консультант)	Эксперт, чьё мнение запрашивается до принятия решения (двусторонняя коммуникация).	
I — Informed (Информируемый)	Тот, кого уведомляют после выполнения задачи (односторонняя коммуникация).	

🔹 Структура матрицы RACI:

Строки — задачи, этапы или deliverables проекта (например, «Согласование ОПР», «Тестирование ПР»).

Столбцы — роли или конкретные участники («Аналитик», «Владелец процесса», «Разработчик», «СППР»).
В каждой ячейке указывается одна или несколько букв R/A/C/I.

🔹 Зачем нужна матрица RACI?

Устраняет неопределённость: кто за что отвечает.
Предотвращает дублирование усилий (несколько «R» без координации).
Избегает пробелов (задача без «R» или «A»).
Ускоряет принятие решений (ясно, кто утверждает).
Упрощает коммуникацию (кого консультировать, кого информировать).

💡 Правила использования:

В каждой строке должен быть хотя бы один R и ровно один A.
Если в задаче много «A» — это риск конфликта.
Если нет «C» — возможно, упущена экспертиза.

✅ Итог:

Матрица RACI — это простой, но мощный инструмент для чёткого распределения ответственности в проекте. В RPA-проектах она особенно важна из-за участия множества стейкхолдеров: владельцев процессов, ИТ-ресурсов, аналитиков, разработчиков и службы поддержки.

13 Что такое проект?

🔹 Определение (по стандарту PMBOK и ГОСТ Р 54869-2011):

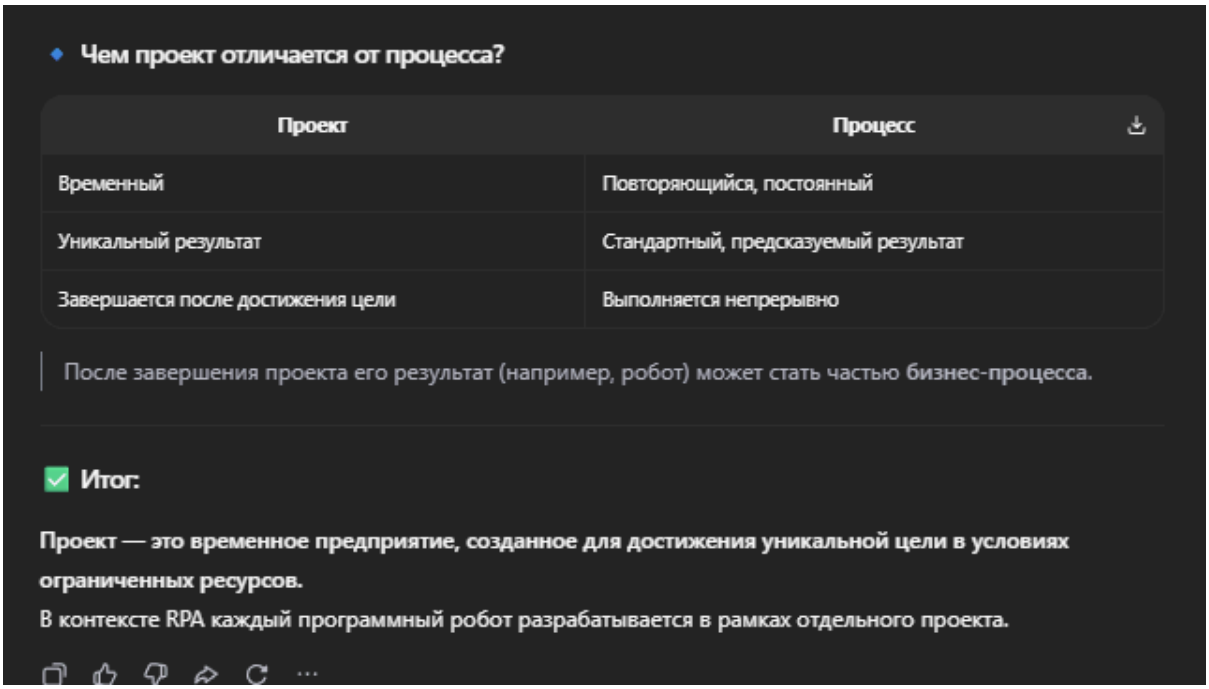
Проект — это временная совокупность взаимосвязанных действий, направленная на создание уникального продукта, услуги или результата.

🔹 Ключевые признаки проекта:

Временность
— У проекта есть чёткие начало и конец.
— Он не является текущей операционной деятельностью.
Уникальность результата
— Проект создаёт что-то новое: систему, процесс, отчёт, робота — то, чего ранее не существовало в таком виде.
Постепенное уточнение (progressive elaboration)
— Детали проекта уточняются по мере продвижения (от идеи к реализации).
Ограниченные ресурсы
— Проект выполняется в рамках заданных сроков, бюджета, персонала и технологий.
Целевая направленность
— Проект реализуется для достижения конкретной цели (например, автоматизация обработки счетов).

🔹 Пример из RPA:

Проект: «Разработка программного робота для автоматической сверки банковских выписок».
— Имеет начало (утверждение Offer-card) и конец (вывод робота в эксплуатацию).
— Создаёт уникальный результат — рабочего робота.
— Ограничен сроками (2 недели), ресурсами (аналитик, разработчик) и бюджетом.



🔹 Чем проект отличается от процесса?

Проект	Процесс
Временный	Повторяющийся, постоянный
Уникальный результат	Стандартный, предсказуемый результат
Завершается после достижения цели	Выполняется непрерывно

> После завершения проекта его результат (например, робот) может стать частью бизнес-процесса.

✅ **Итог:**

Проект — это временное предприятие, созданное для достижения уникальной цели в условиях ограниченных ресурсов.

В контексте RPA каждый программный робот разрабатывается в рамках отдельного проекта.

14 Что такое ограничения проекта

🔹 Определение:

Ограничения проекта — это факторы, которые накладывают рамки на выполнение проекта и определяют условия, в которых команда должна работать. Они ограничивают свободу действий менеджера проекта и влияют на принятие решений.

Источник: Лекция 5, курс «Роботизация бизнес-процессов (RPA) на платформе Атом.РИТА»

🔹 Основные типы ограничений («Треугольник проекта»):

Сроки (Time)

— Фиксированные даты начала и окончания проекта или этапов.

— Пример: «Робот должен быть запущен до конца квартала».

Бюджет (Cost)

— Ограниченные финансовые ресурсы.

— Пример: «На разработку выделено не более 200 тыс. рублей».

Объём работ / Требования (Scope)

— Чётко определённый функционал или результат.

— Пример: «Робот должен обрабатывать только PDF-счета, не Word».

Эти три ограничения взаимосвязаны: изменение одного влияет на другие.

🔹 Дополнительные ограничения (часто актуальны в RPA):

Ресурсы

— Ограниченное число аналитиков, разработчиков, тестировщиков.

— Пример: «В команде только один RPA-разработчик».

Технологии

— Использование только утверждённых платформ (например, только Атом.РИТА).

— Запрет на использование облачных сервисов.

Политики безопасности

— Требования ИБ: запрет на хранение логинов в открытом виде, работа только в изолированной сети.
Юридические и регуляторные требования
— Соответствие ФЗ-152, PCI DSS, ГОСТ и др.
Качество
— Требования к надёжности, точности, времени выполнения.
— Пример: «Робот должен обрабатывать 99% счетов без ошибок».
🔹 Почем inconsistent ограничения — риск проекта
Если ограничения противоречивы (например, «максимальная функциональность за минимальный срок и бюджет»), это создаёт напряжение в проекте и требует:

переговоров с заказчиком,
приоритизации требований (MoSCoW),
компромиссов.
✅ Итог:
Ограничения проекта — это границы возможного, в рамках которых команда должна достичь цели.
Успешное управление проектом — это баланс между ограничениями и гибкость в их согласовании.
В RPA особенно важны ограничения по технологиям, безопасности и стабильности ИТ-систем.

15 Что согласно модели проектного треугольника произойдет, если увеличить содержание, а сроки и стоимость оставить без изменений

🔹 Модель проектного треугольника («Треугольник компромиссов»)
Модель утверждает, что проект определяется тремя взаимосвязанными ограничениями:

Содержание / Объём работ (Scope)
Сроки (Time)
Стоимость / Бюджет (Cost)
Изменение любого одного параметра неизбежно влияет на хотя бы один из двух других.

🔹 Ответ на вопрос:
Если увеличить содержание (объём работ), не меняя сроки и стоимость, то:

Качество результата снизится.

🔹 Пояснение:
Команда должна выполнить больше работы за те же сроки и без дополнительных ресурсов.
Это приводит к:
спешке,
упрощению решений,
пропуску тестирования,

росту числа ошибок,
снижению надёжности и удобства конечного продукта.
💡 В контексте RPA:
Если добавить обработку Word-документов к роботу, который уже обрабатывает PDF, но не дать больше времени или разработчиков, — робот может начать ошибаться, работать нестабильно или не пройти приёмку.

🔹 Альтернатива (если качество нельзя жертвовать):
Чтобы сохранить качество при увеличении объёма, необходимо:

увеличить бюджет (нанять ещё разработчика),
или продлить сроки (выполнить больше за счёт времени).
✅ Итог:
Согласно модели проектного треугольника, увеличение содержания при фиксированных сроках и стоимости неизбежно ведёт к снижению качества. Это классический пример «раздутого scope» (scope creep), который часто приводит к провалу проектов.