

Skillbox

# **Soft-skills и развитие карьеры**

**Architecture Decision Records  
и коммуникация**

**Андрей Гордиенков**

Solution Architect

ABAX

# В прошлом модуле

- Cloud native решения
- 12 факторов «the twelve-factor app»
- Выделение инфраструктурной логики

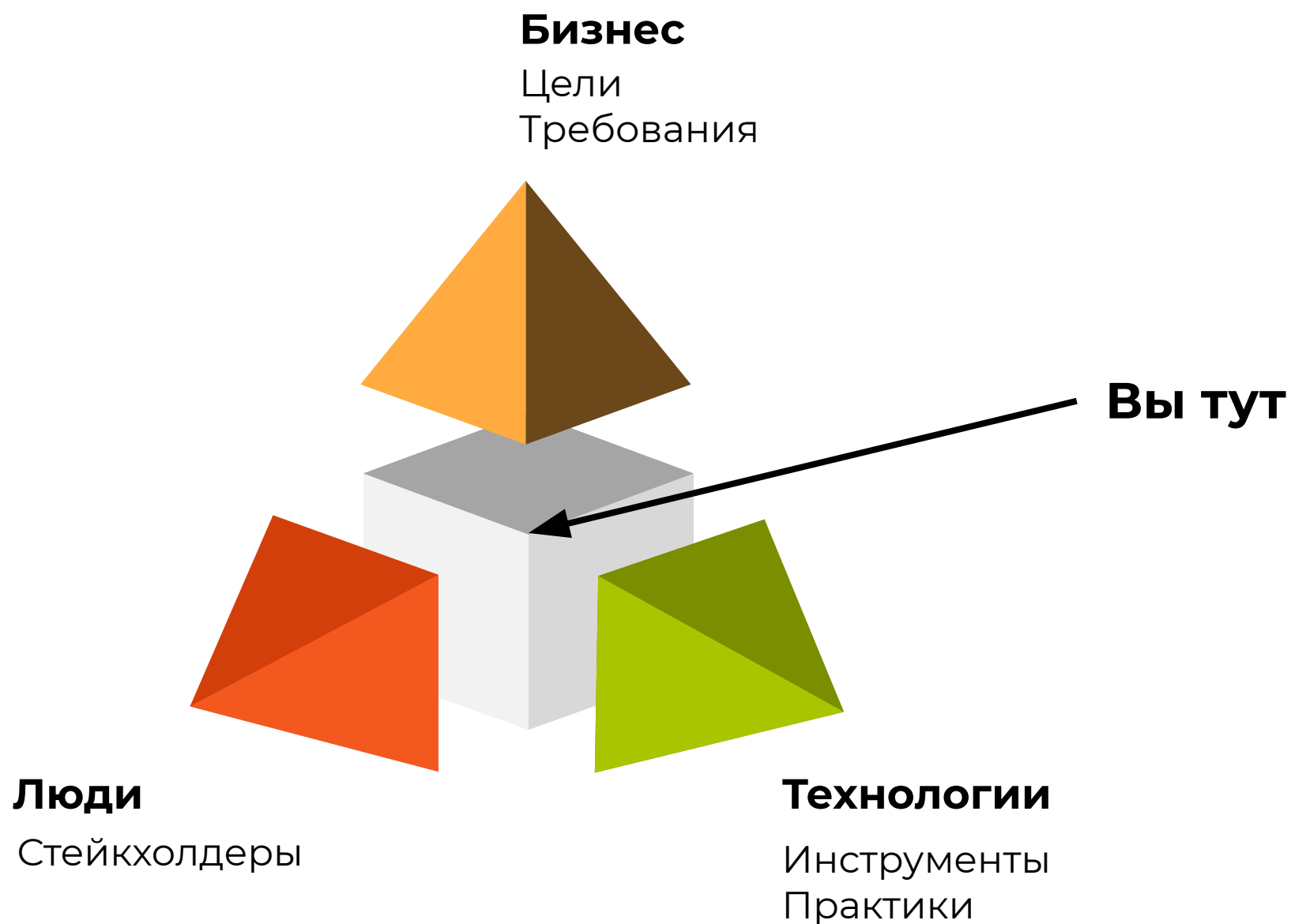
# В этом модуле

- Коммуникация и документирование решений
- Навыки презентации решений
- Работа с командой и саморазвитие
- Ресурсы для роста

# На этом уроке

- Lightweight Architecture Decision Record
- Письменная коммуникация

# Баланс сил



# Задача архитектора — принимать решения

- О решениях надо помнить
- Решения надо документировать



# Architecture Decision Records

Самая сложная вещь в течении проекта — отслеживать причины изменений. Код скажет как, но не скажет почему.

Поэтому без отслеживания причин только две опции:

- принимать решения вслепую
- изменять решения вслепую

# Architecture Decision Records

В 2011 году Michael Nygard предложил описывать решения в коротком формате на странице A4.

1. Название
2. Дата
3. Контекст
4. Решение
5. Статус
6. Последствия

# Architecture Decision Records

В 2011 году Michael Nygard предложил описывать решения в коротком формате на странице A4.

1. Название
2. Дата
3. Контекст
  - Опция 1
  - Опция 2
4. Решение
5. Статус
6. Последствия



# Формат

- Markdown
- Очень малый размер — A4

Основные плюсы:

- возможность хранить в репозитории кода
- поддержка отображения «из коробки»
- лёгкость редактирования
- история изменений

# Формат

47 lines (32 xloc) | 1.97 KB

RawBlame

## System approach

Date: 2020-10-29

### Status

Accepted

### Context

We need to decide what the dominating system approach will be. There are several options like: monolith, microservices, micro-kernels, blackboard.

There are also technical constraints and business drivers even if the sky is blue and we are free of choice, developer's team experience and capabilities should be considered.

### Alternatives

Key map:

- Promotes
- ++ Strongly promotes
- Neutral
- Negative
- -- Strongly negative

	Monolith	Microservices	Micro-kernel	Modularized Monolith
Ease of Deployment	++	--	-	++
Availability	-	++	+	-
Autonomy	--	++	++	+
Traceability	++	-	+	
Performance	++	+	+	++
Modifiability		++	+	+
Maintainability	-	++	+	+
Integrity	++	--		+
Security	-	++	+	++
Scalability	--	++	-	+

### Decision

Based on alternatives in the context of the business needs and a development team we think that modularized monolith is the best option for now. At the same time it opens the possibility to migrate to microservices when needed.

### Consequences

The main benefits come in the development and deployment area. The team could save a lot of resources setting up dev/test/qa/prod environments. The cost of development and cognitive complexity is low level because working on the same code base requires less synchronization meetings.










Extensibility and modifiability with the proposed approach should be on a high level with properly designed facades between modules.

Extra care and discipline should be applied for inter module communication, or it will be just a monolith with a tendency towards "the big ball of mud". Every module should keep subdomain concepts clean and sometimes copy-pasting a part of code is the best approach to avoid hidden dependencies.

- 001 We are using ADR (template).md
- 002 System approach.md
- 003 Tracing and Monitoring Sytem.md
- 004 Health check endpoints.md
- 005 Service readiness checks.md
- 006 Zero trust architecture.md
- 007 Event sourcing usage.md
- 008 At least once delivery for ready to pay order.md
- 009 Rely on payment service provider.md

# Название

- Порядковый номер
- Максимально конкретное название

 001 We are using ADR (template).md
 002 System approach.md
 003 Tracing and Monitoring Sytem.md
 004 Health check endpoints.md
 005 Service readiness checks.md
 006 Zero trust architecture.md
 007 Event sourcing usage.md
 008 At least once delivery for ready to pay order.md
 009 Rely on payment service provider.md

Никогда не удалять документы и кардинально не менять содержания.  
Меняться может статус.

# Контекст

Описание причин, подводящих к необходимости принятия решения.  
Описание ограничений и предположений для описываемой ситуации.

Рекомендации:

- узкий контекст
- описать все предположения
- не склонять к решению
- опции описывать в этой секции

# Контекст

## Опции:

### Alternatives

Key map:

- ◦ Promotes
- ++ Strongly promotes
- ○ Neutral
- ◦ Negative
- -- Strongly negative

	Monolith	Microservices	Micro-kernel	Modularized Monolith
Ease of Deployment	++	-	-	++
Availability	-	++	+	-
Autonomy	--	++	++	+
Traceability	++	-	+	○
Performance	++	+	+	++
Modifiability	○	++	+	+
Maintainability	-	++	+	+
Integrity	++	--	○	+
Security	-	++	+	++
Scalability	--	++	-	+

# Решение

Описать решение максимально кратко, в 2–3 предложения. Описание в настоящем времени. Может быть ссылка на выбранную опцию, описанную в предыдущей секции.

# Статус

Описание статуса решения:

- отложено
- принято
- отвергнуто

Очень важно документировать отвергнутые решения, чтобы не возвращаться многократно к их обсуждению.

# Последствия

Последствия принятого (или нет) решения с описанием:

- позитивных моментов решения
- негативных моментов решения
- возможных рисков, связанных с решением



# О чём важно помнить

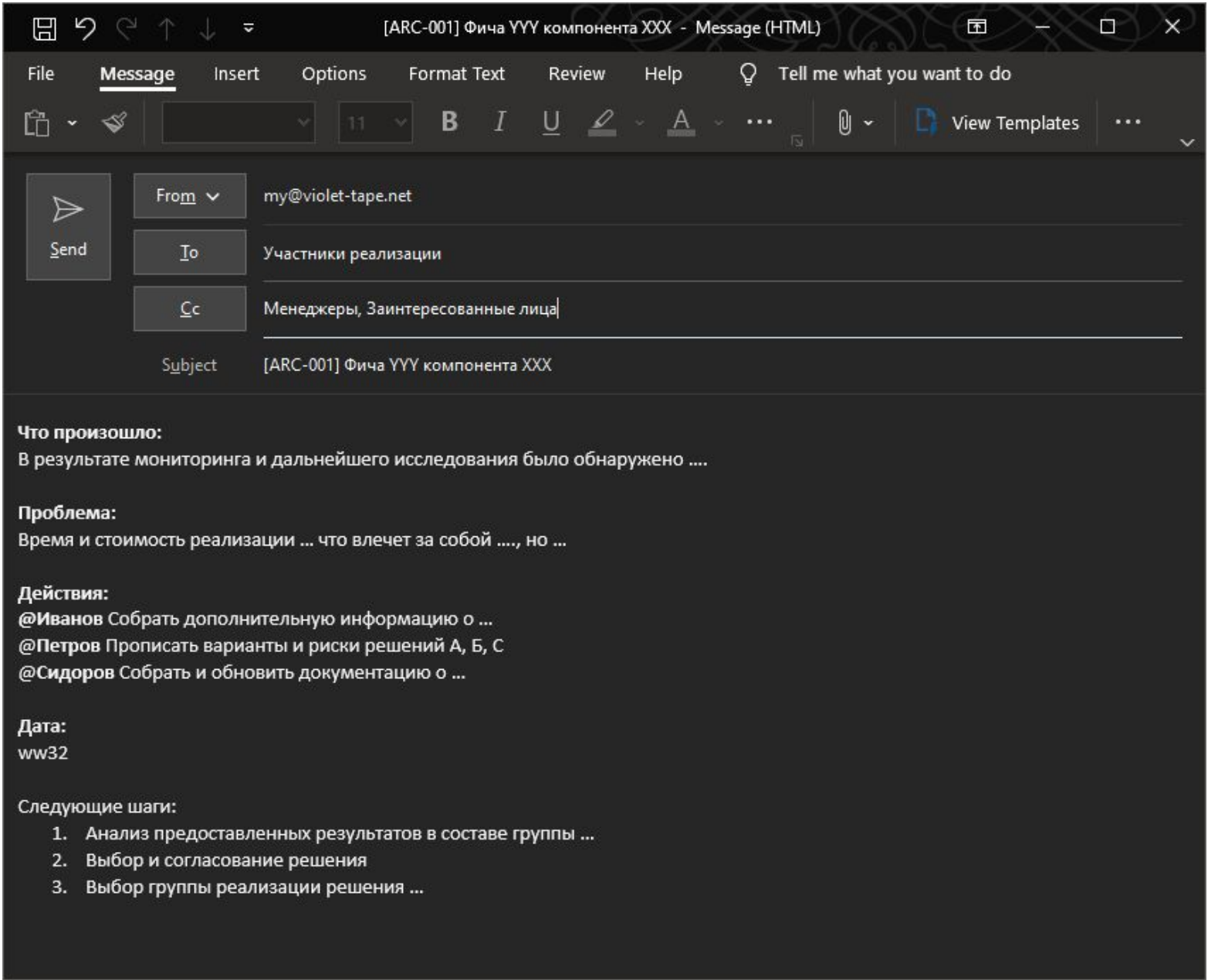
- Очень часто ограничения системы записывают в ADR
- Требования тоже не попадают в раздел ADR
- При документации ADR должна быть реальная альтернатива и выбор. Именно поэтому есть секция «Последствия»
- При выборе подходов и инструментов важно описать анализ опций
- Запись отвергнутых опций может быть даже важнее принятых
- В описании делать акцент на причины принятия решения, а не на детали реализации

# Коммуникация

Старайтесь следовать общей концепции ADR

- Тема
- [Проект] Описание проблемы
- Контекст
- Проблема
- Шаги для решения с упоминанием исполнителей
- Срок решения
- Ожидаемый результат и следующие шаги

# Коммуникация



# Коммуникация

- При пересылке письма новым участникам делайте краткое резюме предыдущей беседы, если сообщений было много
- Прописывайте, что вы ожидаете от нового участника:
  - информирование
  - предоставление информации
  - конкретные действия
- Перечитывайте написанное (не осталось ли в вашей голове контекста письма)
- Оформляйте ссылки на упомянутые артефакты
- Описывайте назначение приложенных артефактов

# Выводы

- ADR позволяют информированно и последовательно принимать решения в проекте
- ADR решают проблему передачи информации в коллективе
- ADR описывают причины принятия решений
- При написании документации думайте о времени других и как помочь другим решить задачи, необходимые вам
- Документация нужна для эффективной коммуникации между участниками, а не для самой документации

# Что дальше?

- Описание архитектуры
- Презентация архитектуры

Skillbox

**Спасибо  
за внимание!**