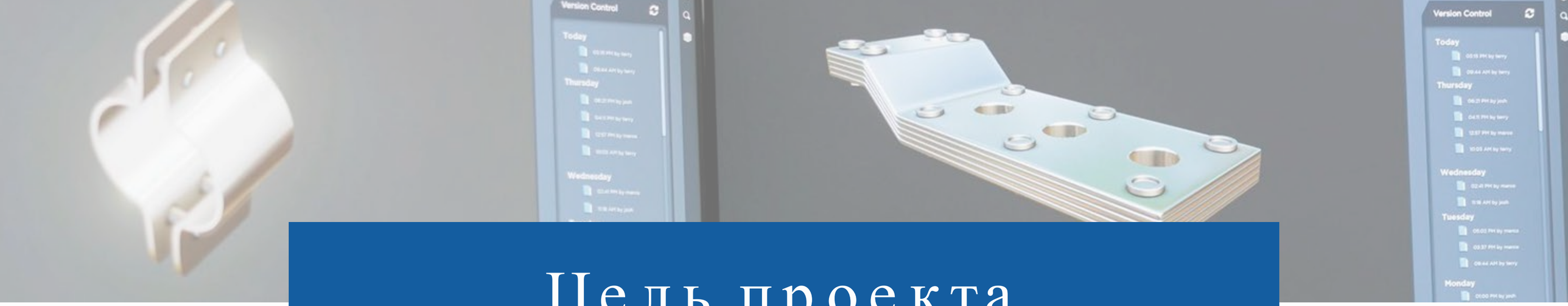
CAD AI

Генерация моделей

Команда студентов ВГТУ | Апрель 2025



Цель проекта

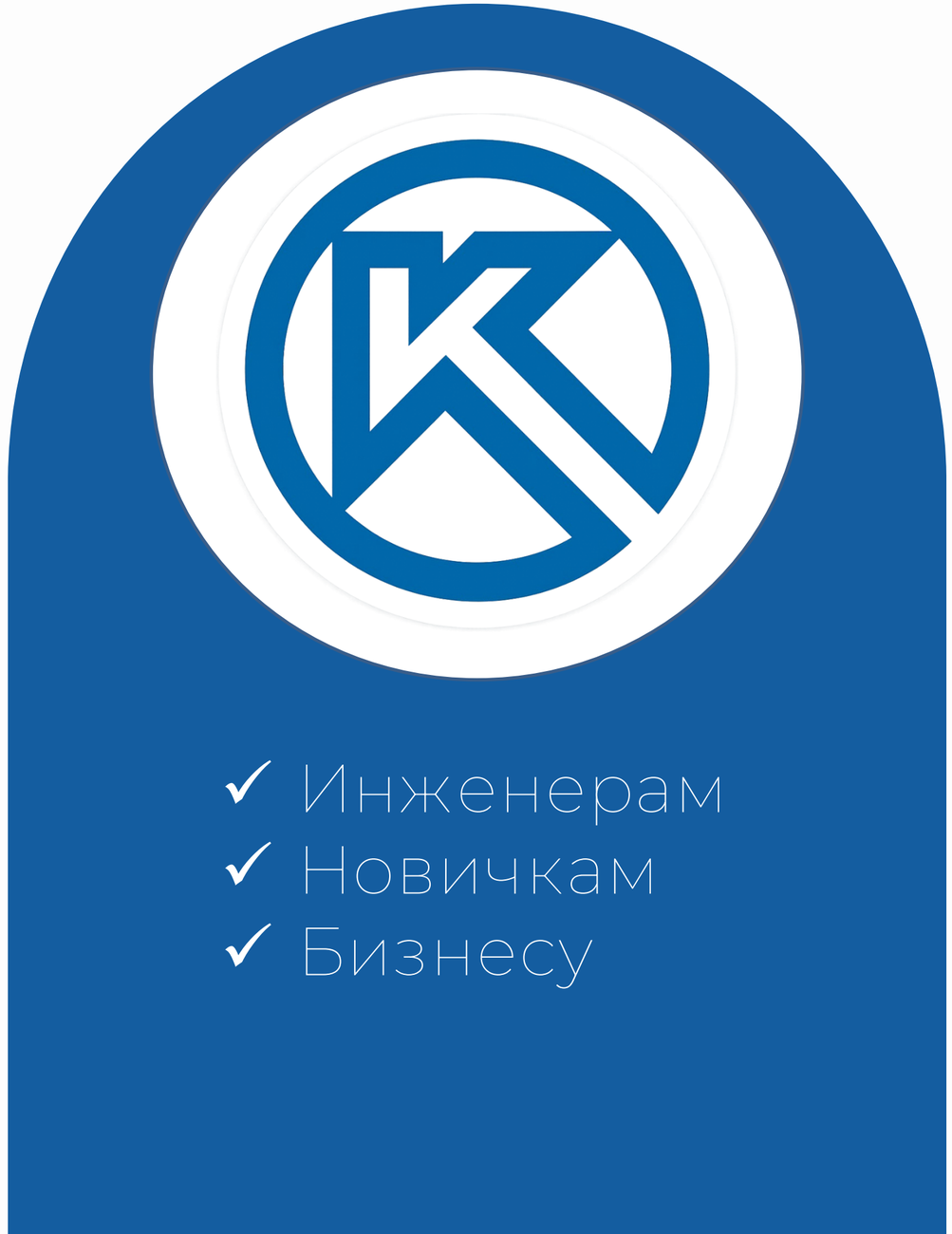
CAD AI — интеллектуальное расширение возможностей KOMPAS-3D. Целью проекта CAD AI является создание интеллектуальной системы, способной автоматически генерировать трёхмерные модели на основе текстовых описаний, вводимых пользователем. Мы хотим сократить время проектирования, упростить работу инженеров и снизить порог вхождения для пользователей, не имеющих большого опыта в CAD-среде.

Зачем это нужно?

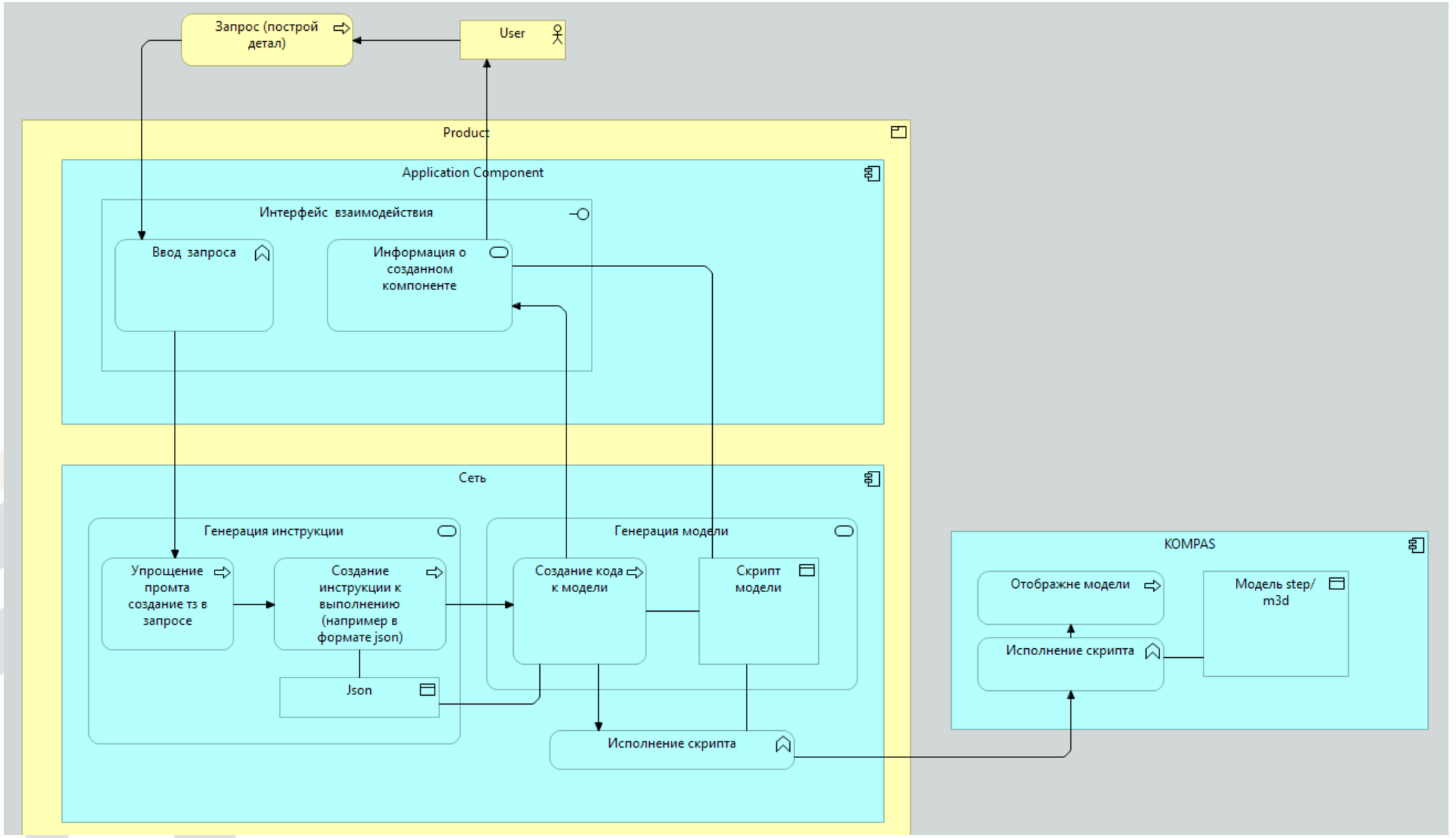
1. Ускорение процесса 3D-проектирования
2. Минимизация рутинной работы для инженеров и проектировщиков
3. Снижение порога входа в работу с CAD для новых пользователей
4. Интеграция ИИ в существующие САПР-платформы

Как это работает?

Пользователь вводит текстовый запрос
→ нейросеть интерпретирует его в инструкцию
→ система автоматически строит 3D-модель в KOMPAS-3D



- ✓ Инженерам
- ✓ Новичкам
- ✓ Бизнесу


Запрос (построй деталь)
User
Product
Application Component
Интерфейс взаимодействия
Ввод запроса
Информация о созданном компоненте
Сеть
Генерация инструкции
Упрощение промта создание тз в запросе
Создание инструкции к выполнению (например в формате json)
Json
Генерация модели
Создание кода к модели
Скрипт модели
Исполнение скрипта
KOMPAS
Отображне модели
Модель step/ m3d
Исполнение скрипта

Этапы работы

AI-блок: YandexGPT (или другая модель) → формирует инструкции

Интерпретатор: собственный код на Python → преобразует инструкции в команды

Kompas SDK: исполняет команды → визуализирует модель

01

Пользователь вводит текстовый запрос

Пример: *«Создай цилиндр диаметром 50 мм и высотой 30 мм»*

02

Интерфейс взаимодействия

Получение запроса;
Отображение информации о процессе/результате

03

Нейросеть (через API)

Обработка запроса
Генерация инструкций в формате DSL/JSON

04

Интерпретация инструкций

Преобразование команд в вызовы KOMPAS SDK (через Python-скрипт)

05

Запуск построения в KOMPAS

Выполнение макроса
Отображение готовой 3D-модели

Вариации исполнения: Текст →DSL →Интерпретатор

- Вариант 1 отправляет подготовленный текстовый запрос в YandexGPT
- Полученная DSL-инструкция — это упрощённая последовательность команд, например:

```
create_sketch()
draw_rectangle(100, 50)
extrude(20)
```

- DSL обрабатывается интерпретатором, который вызывает соответствующие методы Kompas SDK

Преимущества

1. Простой и понятный синтаксис;
2. Легко логировать и тестировать;
3. Высокая прозрачность и управляемость.

Проблемы

1. Менее гибкий формат;
2. Труднее описывать сложные зависимости и параметры.

Вариации исполнения: Текст →JSON →Интерпретатор

- Вариант 2 также работает с текстовыми запросами, но получает от нейросети структурированный JSON:

```
{
  "instructions": [
    {
      "operation": "draw_rectangle",
      "parameters": {"width": 100, "height": 50}
    },
    {
      "operation": "extrude",
      "parameters": {"depth": 20}
    }
  ]
}
```

- Интерпретатор парсит JSON и выполняет команды через Kompas SDK

Преимущества

1. Более гибкий и масштабируемый формат;
2. Удобен для сложных параметризованных построений;
3. Проще валидировать и обрабатывать.

Проблемы

1. Более громоздкий формат;
2. Требует строгой структуры и проверки входных данных.

⚖️ Сравнение форматов:

Параметр	DSL	JSON
Простота	✅ Высокая	⚠️ Средняя
Гибкость	⚠️ Низкая	✅ Высокая
Расширяемость	⚠️ Ограниченная	✅ Отличная
Поддержка параметров	⚠️ Базовая	✅ Полная

Получение DSL инструкций для построения моделей

```python
prompt_codeBuild = {
    "modelUri": "gpt://b1g6o7ju6g5hrv2h5j0n/yandexgpt",
        "completionOptions": {
            "stream": False,
            "temperature": 0.6,
            "maxTokens": "2000"
        },
        "messages": [
            {"role": "system", "text": "Ты помощник, который принимает пошаговый алгоритм построения и преобразует его в код для построения графических примитивов. Используй только следующие к
            {"role": "user", "text": result_Algoritm},
    ]
}

response = requests.post(url, headers=headers, json=prompt_codeBuild)
result_codeBuild = response.json()

try:
    result_codeBuild = result_codeBuild["result"]["alternatives"][0]["message"]["text"]
    print(result_codeBuild)
except KeyError:
    print("Ошибка обработки ответа:", json.dumps(result_codeBuild, indent=2))
```

[86] Python

```
Circle(0, 0, 50)
LineSeg(25, 25, 75, 25)
LineSeg(75, 25, 75, 75)
LineSeg(75, 75, 25, 75)
LineSeg(25, 75, 25, 25)
LineSeg(-25, -25, -75, -25)
LineSeg(-75, -25, -75, -75)
LineSeg(-75, -75, -25, -75)
LineSeg(-25, -75, -25, -25)
```

Пример JSON инструкций, полученных от AI

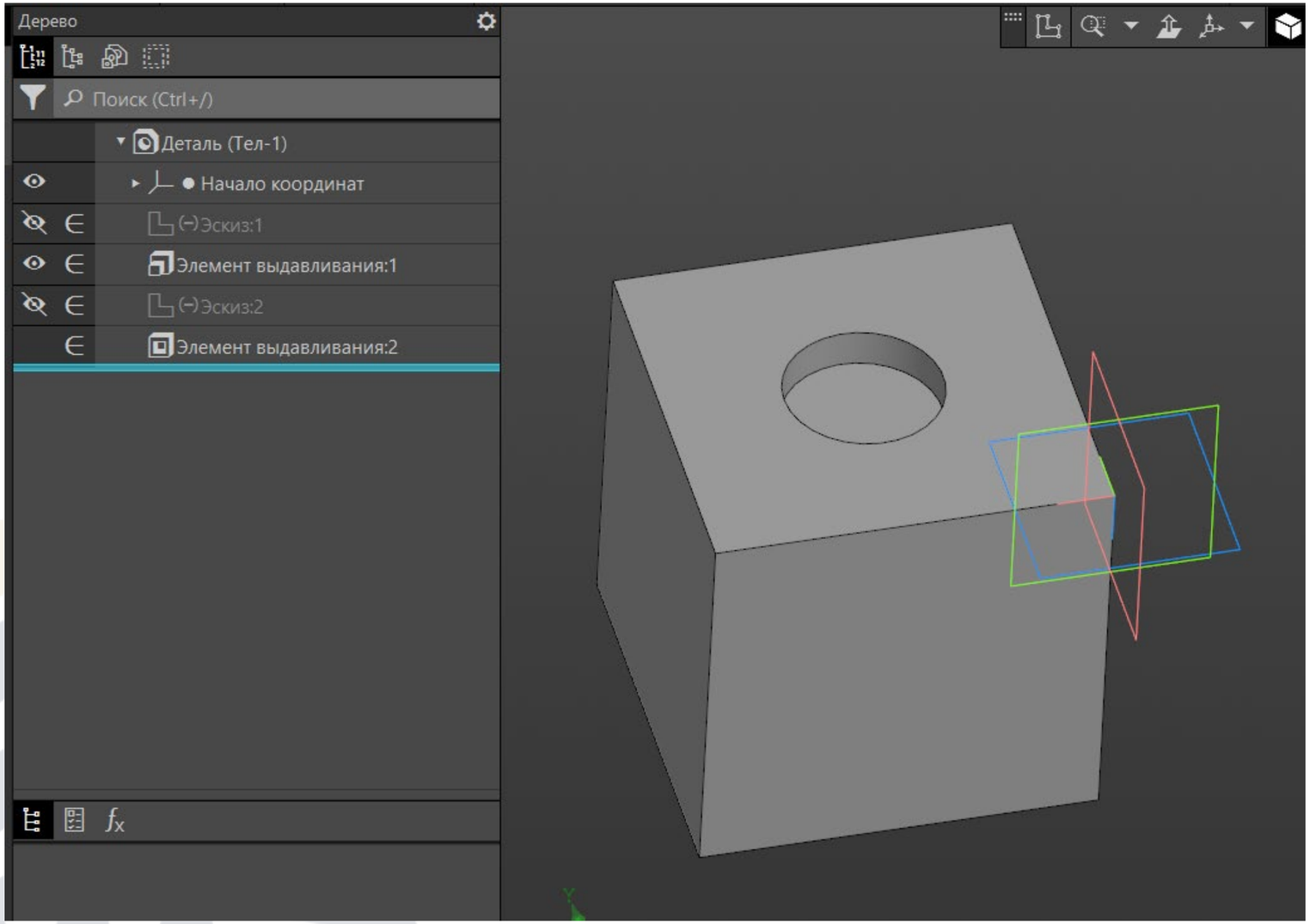
```
# Пример JSON для построения куба с отверстием
json_instruction = """
{
    "steps": [
        {"action": "sketch", "plane": "XOY", "entities": [
            {"type": "line", "start": [0, 0], "end": [100, 0]},
            {"type": "line", "start": [100, 0], "end": [100, 100]},
            {"type": "line", "start": [100, 100], "end": [0, 100]},
            {"type": "line", "start": [0, 100], "end": [0, 0]}
        ]},
        {"action": "extrude", "height": 100},
        {"action": "sketch", "plane": "XOY", "entities": [
            {"type": "circle", "center": [50, 50], "radius": 20}
        ]},
        {"action": "cut", "depth": 1000},
        {"action": "color", "value": [128, 128, 255]}
    ]
}
"""
```

Класс-интерпретатор (JSON-вариация)

```python
class Kompas3DBuilder:
    def __init__(self):
        self.iPart = iPart
        self.last_sketch = None

    def start_sketch(self, plane):
        """Создание нового эскиза на заданной плоскости."""
        sketch = self.iPart.NewEntity(kompas6_constants_3d.o3d_sketch)
        definition = sketch.GetDefinition()
        plane_obj = self.iPart.GetDefaultEntity(
            getattr(kompas6_constants_3d, f"o3d_plane{plane}")
        )
        definition.SetPlane(plane_obj)
        sketch.Create()
        self.last_sketch = sketch
        return definition.BeginEdit()

    def add_line(self, doc2d, x1, y1, x2, y2):
        """Добавление линии в эскиз."""
        doc2d.ksLineSeg(x1, y1, x2, y2, 1)

    def add_circle(self, doc2d, x, y, radius):
        """Добавление окружности в эскиз."""
        doc2d.ksCircle(x, y, radius, 1)

    def finish_sketch(self):
        """Завершение эскиза."""
        if not self.last_sketch:
            raise RuntimeError("Эскиз не найден, невозможно завершить редактирование.")
        self.last_sketch.GetDefinition().EndEdit()
```

```python
def process_json(self, json_data):
    """Обработка JSON-инструкций для создания модели."""
    for step in json_data.get("steps", []):
        match step.get("action"):
            case "sketch":
                plane = step.get("plane")
                doc2d = self.start_sketch(plane)
                for entity in step.get("entities", []):
                    match entity.get("type"):
                        case "line":
                            self.add_line(doc2d, *entity["start"], *entity["end"])
                        case "circle":
                            self.add_circle(
                                doc2d, *entity["center"], entity["radius"]
                            )
                self.finish_sketch()
            case "extrude":
                self.extrude(step.get("height"))
            case "cut":
                self.cut(step.get("depth"))
            case "color":
                self.apply_color(step.get("value"))
            case _:
                raise ValueError(f"Неизвестная команда: {step.get('action')}")
```

Пример модели, созданной по текстовому промпту



Идея, от которой мы отказались

Использование сторонних AI моделей

01 Быстрая и простая работа через API

02 Хорошее понимание естественного языка

03 Не требует подготовки датасета и больших мощностей

04 Не обучены на CAD-доменных данных

05 Генерация может быть неточной или неконсистентной

06 Сложно обучить на CAD-командах, улучшить точность

Идея, от которой мы отказались

Генерация готовых моделей (STEP, M3D, STL и др.)

01 Быстрая визуализация без построения через SDK

02 Не зависит от SDK и прочих инструментов, может обрабатывать запросы удалённо

03 Полученные файлы часто полигональные, не пригодны для правок

04 Зависимость от форматов

05 Более долгие вычисления и выше требования по мощностям

06 Потеря истории построений и параметричности

Проблема

Векторные и топологические представления, проблемы AI

01 Нейросети плохо работают с геометрическими зависимостями, эскизами, B-rep

02 Сложности с пространственным мышлением (пересечения, массивы, грани и т.д.)

03 Невозможность учёта эскизных плоскостей и ограничений

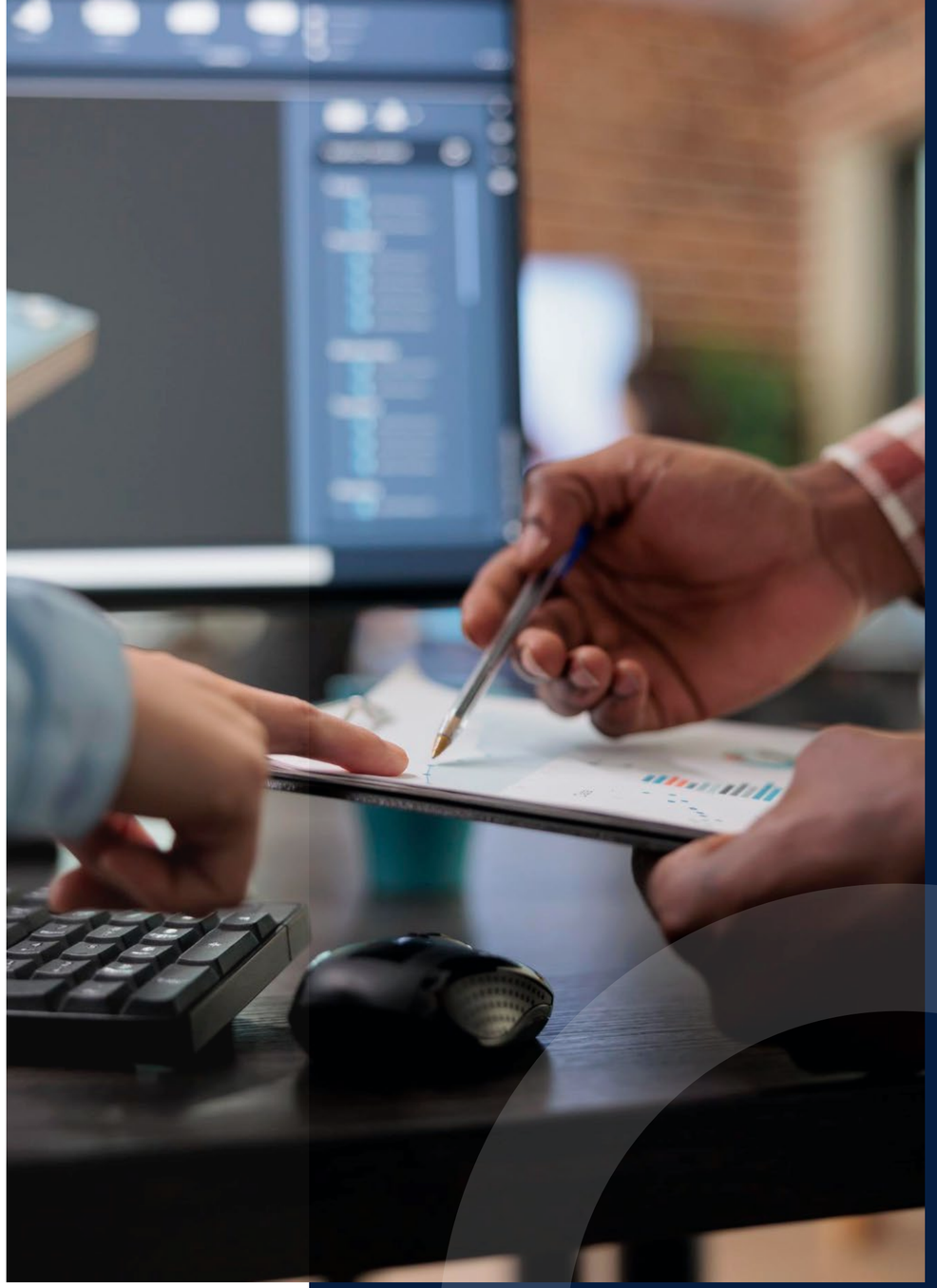
04 Неустойчивость к некорректным данным

Проблемы

1. Неполные или некорректные запросы
   - Пользователь может не указать все параметры (размеры, форму, плоскость)
   - Нейросеть не всегда уточняет, а «додумывает» сама
2. Генерация ошибочных или невыполнимых инструкций
   - Некорректный порядок команд (например, extrude без эскиза)
   - Ошибки синтаксиса, несоответствие формату DSL/JSON
3. Ограниченность возможностей макросов
   - Некоторые операции невозможно выразить простыми командами
   - Отсутствует доступ к полноценной геометрии тела (например, для анализа пересечений)
4. Трудности в интерпретации
   - Интерпретатор должен учитывать контекст, эскизы, параметры
   - Ошибки в логике могут не обнаружиться до исполнения в Kompas
5. Отсутствие обратной связи
   - Невозможно понять, что именно пошло не так при ошибке
   - Нет полноценной диагностики в случае сбоя исполнения

Пути решения

1. Введение предварительной проверки и уточнения
   - Проверка наличия всех нужных параметров перед генерацией кода
   - Возможность уточнения недостающих данных у пользователя
2. Слой валидации и отладки перед запуском
   - DSL/JSON проходят через валидацию
   - Тестовый «dry run» без запуска в Kompas
3. Расширение интерпретатора
   - Поддержка дополнительных геометрических примитивов
   - Работа с массивами, сложными операциями
4. Переход на более мощный API (Kompas API напрямую)
   - Повышение гибкости и точности построений
   - Расширение возможностей за пределы макросов
5. Логирование, отладка и визуализация
   - Отображение промежуточных шагов
   - Сохранение логов для анализа сбоев



Спасибо за внимание!