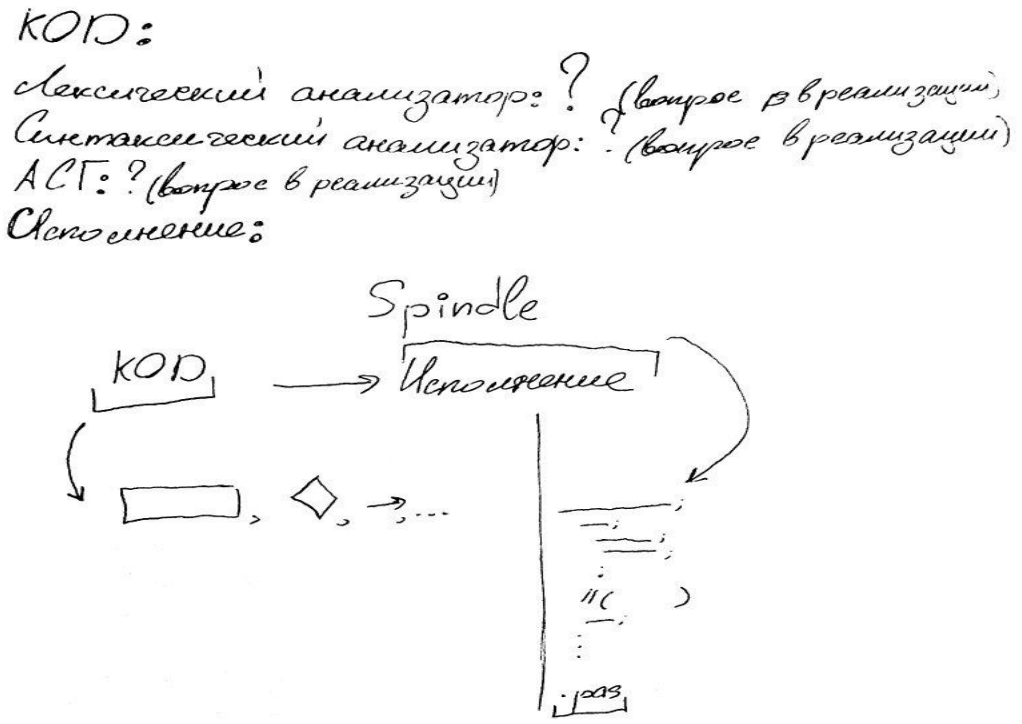
Идея разбить всю реализацию на стандартные единицы написания ЯП: код (в нашем случае графические формы), лексический анализатор, синтаксический анализатор, АСГ (абстрактный семантический граф), исполнение (исходный код на Pascal), т.к., в целом, можно сказать, что мы разрабатываем своего рода интерпретатор для языка UML или же блок-схем в код на Pascal.
В дополнение: мне кажется целесообразным выбрать в качестве стандарта графической части UML, т.к. последний обладает богатой документацией и является более гибким. Из явных минусов UML можно выделить лишь его относительную сложность в освоении в сравнении с блок-схемами.



1. **Лексический анализатор (лексер)**

Wikipedia: в информатике лексический анализ («токенизация», от англ. tokenizing) — процесс аналитического разбора входной последовательности символов на распознанные группы — лексемы, с целью получения на выходе идентифицированных последовательностей, называемых «токенами» (подобно группировке букв в словах). В простых случаях понятия «лексема» и «токен» идентичны, но более сложные токенизаторы дополнительно классифицируют лексемы по различным типам («идентификатор, оператор», «часть речи» и т. п.). Лексический анализ используется в

компиляторах и интерпретатора хисходного кода языков программирования, и в различных парсерах слов естественных языков.

В качестве исходной информации будет подаваться доска со структурой (под структурой будем понимать схему на языке UML или блок-схем), которую лексер будет разбирать на отдельные лексемы. Здесь важно правильно определить вид одной отдельной конечной лексемы. Так, например, в текстовых ЯП лексеры выделяют в токены отдельные операторы, переменные, ключевые слова и т.д. Конечно же мы тоже можем прибегнуть к подобной реализации: каждому элементу (в отдельных случаях группам элементов - циклы) поставить в соответствие некоторое слово - название операции или имя операнда. В ходе дальнейшего изучения и разработки возможно изменение этой концепции в сторону более гибкой и простой системы (я на это надеюсь).

2. **Синтаксический анализатор (парсер)**

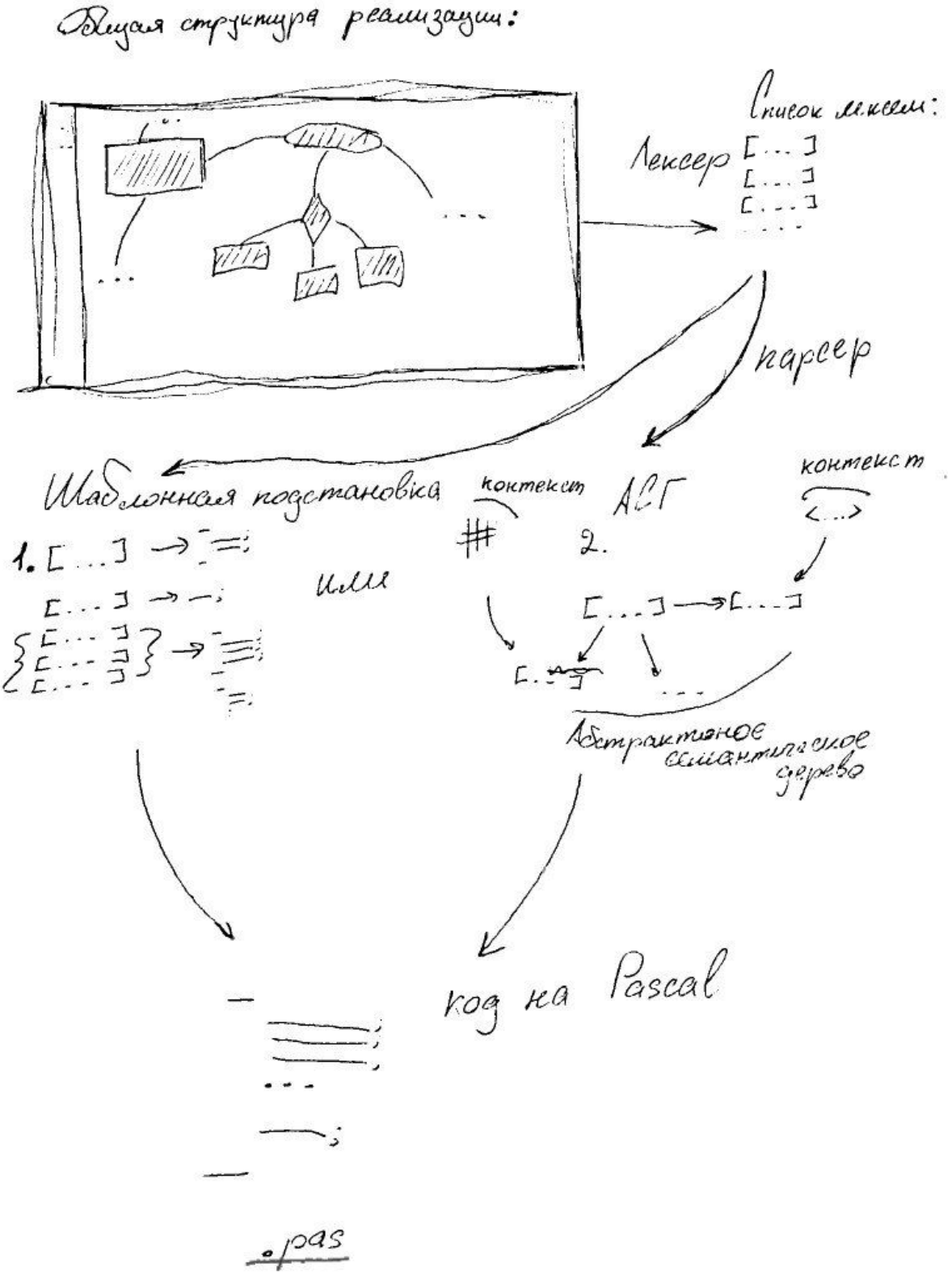
Wikipedia: в лингвистике и информатике — процесс сопоставления линейной последовательности лексем (слов, токенов) естественного или формального языка с его формальной грамматикой. Результатом обычно является дерево разбора (синтаксическое дерево).

Собственно именно парсер будет сопоставлять структуре код на Pascal. Вопрос в том, как именно это будет происходить: либо шаблонно подставлять куски кода для каждой конкретной ситуации(отвратная масштабируемость, трудоемкость внесения всех случаев, явно что-то пропустим), либо создавать АСГ и на его основе генерировать код(пока что темный лес, а не вариант, но чувствую у него есть перспектива - будет дальнейшее изучение).

3. **АСГ (абстрактный семантический граф)**

Полность перепощена статья с Wikipedia: в информатике абстрактным семантическим графом (АСГ) называется структура данных, используемая для представления или извлечения семантики выражения на формальном языке (например, на языке программирования).
Абстрактный семантический граф — это более высокий уровень абстракции, чем абстрактное синтаксическое дерево (АСД), которое используется для описания синтаксической структуры выражения или программы.
Абстрактный семантический граф обычно конструируется из абстрактного синтаксического дерева процессом обогащения и абстрагирования. Обогащением может быть, например, добавление обратных указателей, рёбер из вершины идентификатора (где используется переменная) в вершину, соответствующую

объявлению этой переменной. Абстрагирование может включать удаление деталей, которые нужны только для синтаксического, но не семантического анализа.



Нужно также обсудить интерфейс самой среды разработки, используемый инструментарий.