GPSS и ПОСП

– примеры реализации

проблемно-ориентированного языка

(domain-specific language, DSL)

domain-specific language, DSL

*(**варианты определения**:*

предметно-ориентированные,

предметно-специфические,

проблемно-ориентированные)

Языки общего назначения:

Универсальные;

Мощные;

Избыточные

Предметно-ориентированные языки:

Узкие;

Ограниченной мощности;

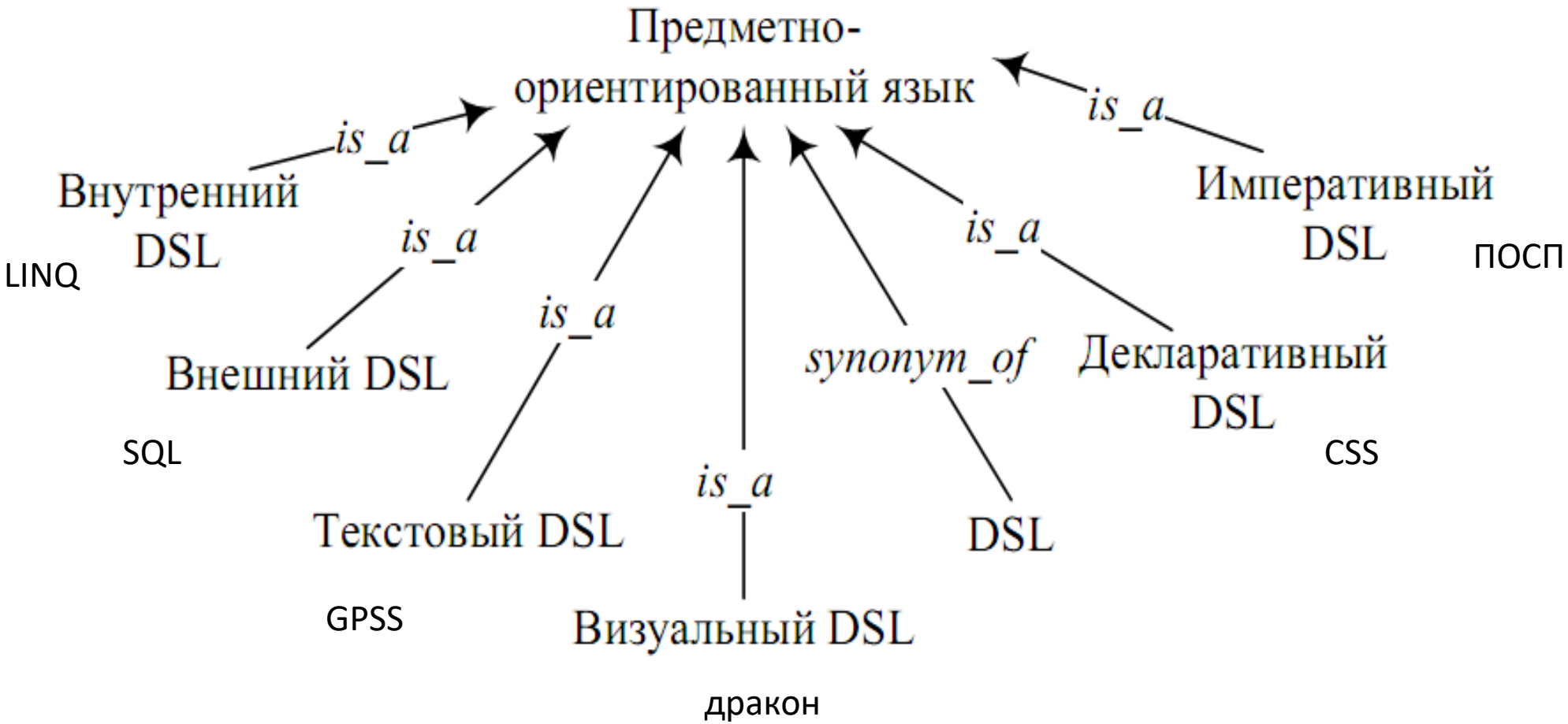
Компактные

“Предметно-ориентированный язык - это язык программирования с ограниченными выразительными возможностями, ориентированный на некую конкретную предметную область” - *Мартин Фаулер*

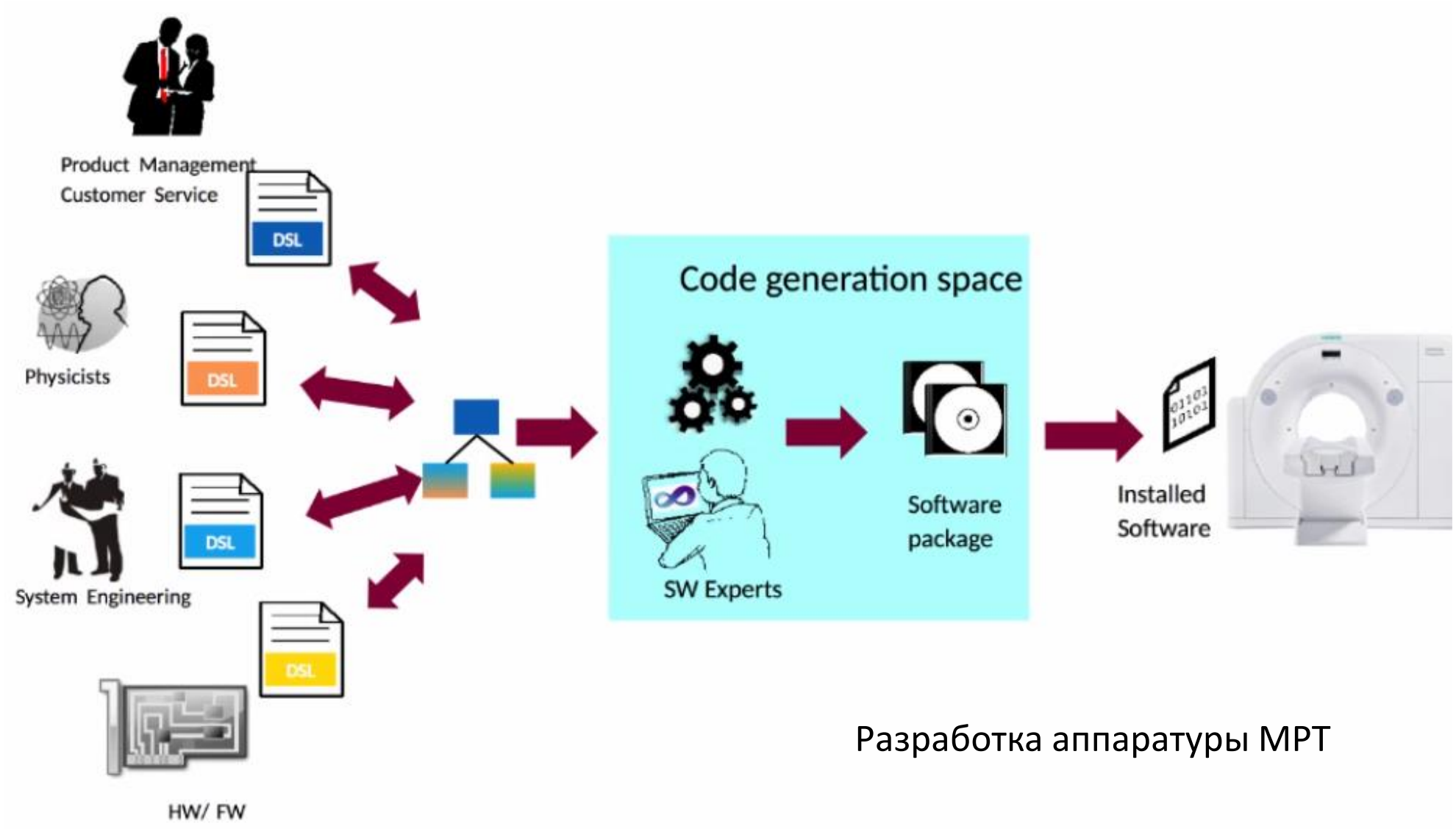
Что такое D S L

- Язык описания предметной области или определенной проблематики
- Предназначен для описания решения задачи в терминах и концепциях, приближенных к предметной области
- Может использоваться приемлемая для конкретной предметной области нотация – от графических схем до описания на естественном языке

Что такое D S L



Что такое D S L



Разработка аппаратуры МРТ

JetBrains https://www.youtube.com/watch?v=pVIywLXDuRo

Основная идея спец. языка:

- ✓ предметно/объектно - ориентированный подход
- ✓ компонентное программирование
- ✓ структурный базис для узкоспециализированных задач
- ✓ формализация описания предметной области

<u>основные типы DSL</u>

- внутренний (встраиваемый, internal) DSL;
- внешний DSL (external) — т.е. написан на языке, отличающемся от основного языка программирования.

Есть интегрированные среды разработки DSL (Language Workbench, MetaCASE)

Встраиваемые DSL / текстовые /

- ✓ Реализуются средствами языка программирования
- ✓ Не требуют специального компилятора
- ✓ Просты в реализации
- ✓ Выразительные средства:
  - ❖ API специальных библиотек (например, .NET)
  - ❖ Объектная модель и интерфейсы работы с ней
  - ❖ Макросы (как в C++)
  - ❖ Специализированные комментарии (типа JavaDoc)
  - ❖ Аннотации (как в C#)
  - ❖ Расширения языка (типа LINQ)

Внешние DSL

текстовые или графические языки

- ✓ Используется полностью новый синтаксис
- ✓ Может иметь стандартизованный формат (XML)
- ✓ Реализует расширение или специализацию UML
- ✓ Необходим специальный транслятор

[но нет ограничений на выразительные средства]

Методология MetaCASE

Продукты MetaCASE представляют собой узкоспециализированные среды разработки приложений, которые создают пользовательский инструментарий на основе высокоуровневого описания требуемых инструментов.

Технология MetaCASE позволяют определять и создавать инструменты CASE, которые поддерживают различные методологии. Средство настройки CASE сначала определяет желаемую методологию и настраивает соответствующий инструментарий CASE, а разработчики ПО используют этот инструмент CASE для разработки программных систем.

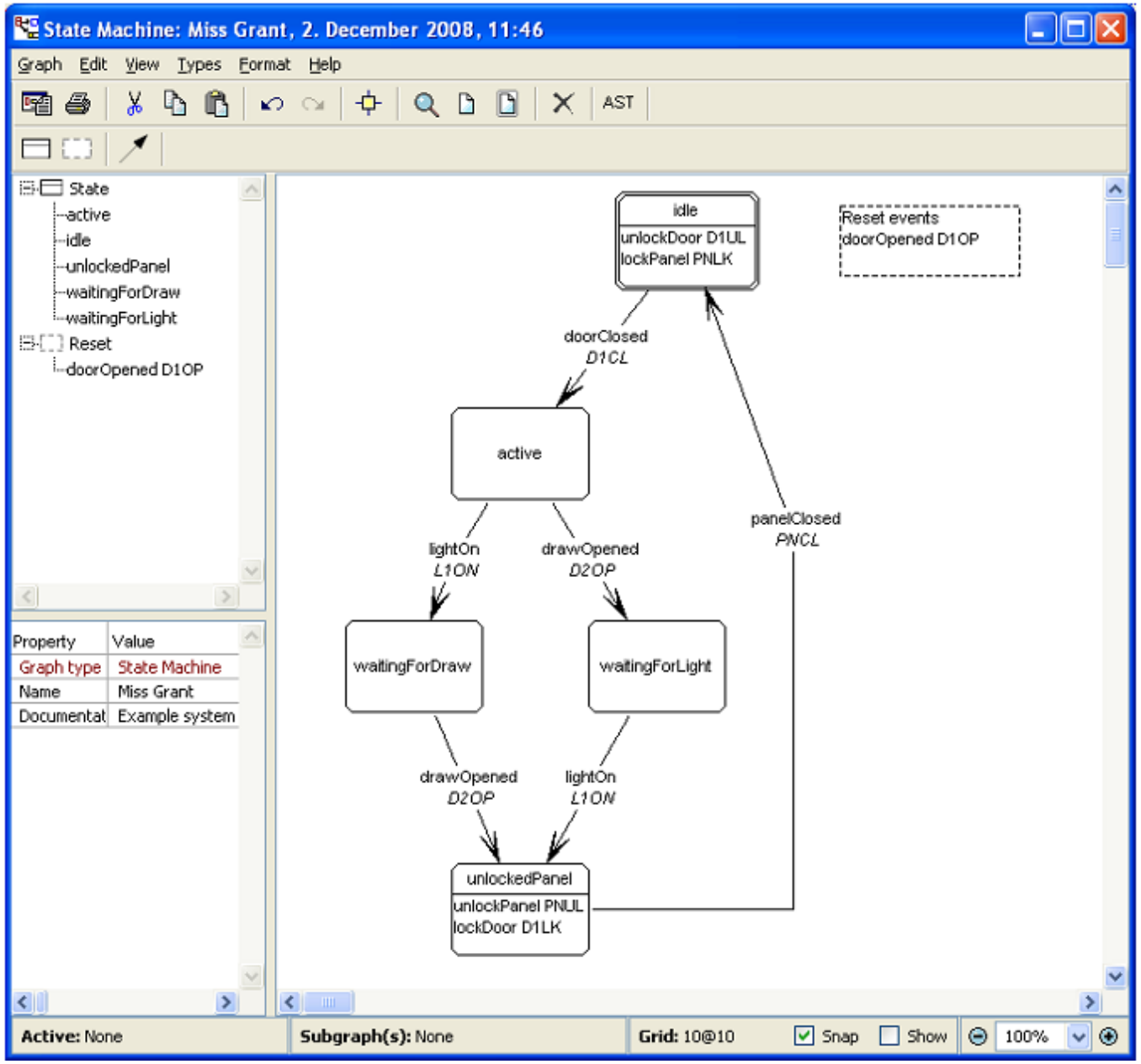
Преимущество этого подхода заключается в том, что один и тот же инструмент используется с разными методологиями, что уменьшает и кривую обучения, и стоимость.

Эту технологию можно использовать в качестве практического инструмента обучения с сокращённой продолжительностью разработки и изучения.

[CASE :: computer-aided software engineering]

Diagramming Tools; Report Generators; Analysis Tools; Central Repository; Documentation Generators; Code Generators

среда разработки DSL (Language workbench MetaEdit)



Примеры metaCASE

DOME

GME

MetaEdit+

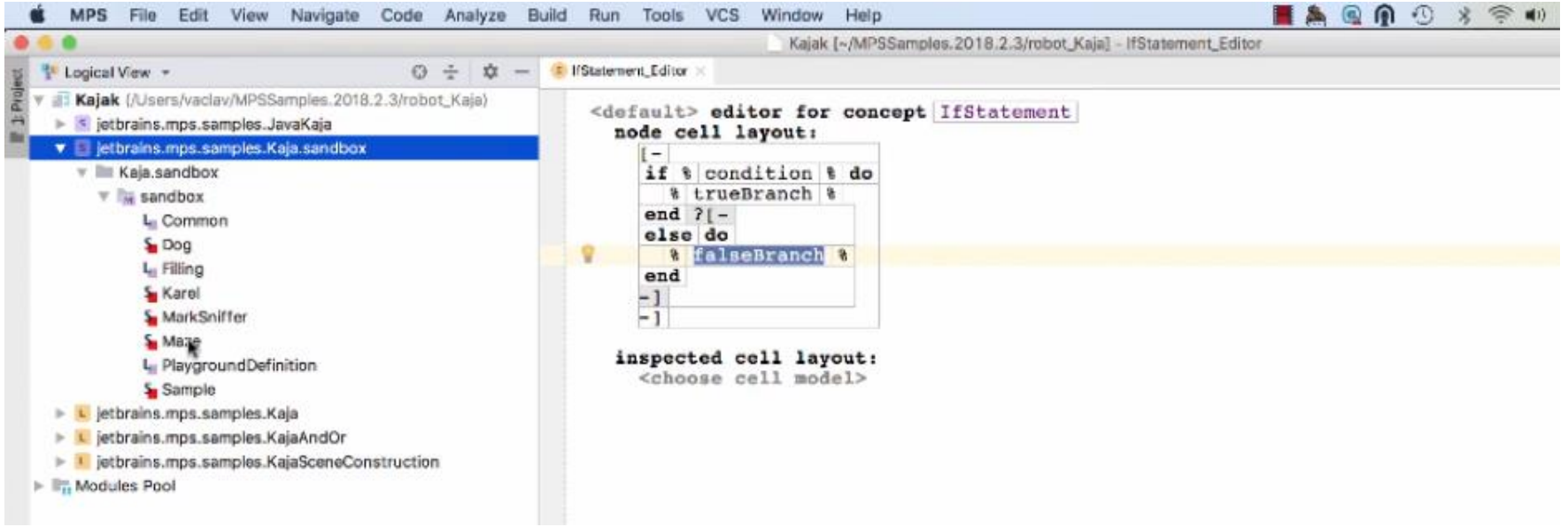
Obeo Designer

Whole Platform

ConceptBase

Jetbrains MPS

среда разработки DSL (Language workbench JetBrains MPS)



С помощью MPS можно самостоятельно создавать редакторы DSL для упрощения использования нового языка.

Специалисты в своих профессиональных областях (не обязательно программисты) смогут работать с этой программной средой, используя язык, созданный на основе специализированной терминологии конкретной предметной области.

Области применения DSL

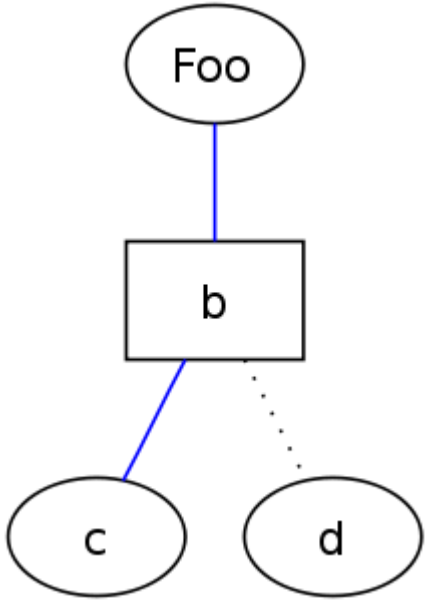
- Описание преобразования текстов (XSLT, Regular-Expresion)
- Описание графов (DGML, GraphML, DOT)
- Описание формул (MathML)
- Оформление документации (DITA, DocBook, Doxygen, markdown)
- Описание форматирования текста (TeX)
- Задание правил для экспертных систем (Drools)
- Задание спецификаций ПО (ACSL)
- Описания правил тестирования (JUnit)
- Model Driven Development (Finite-State-Machine, SCXML)
- Описание архитектур приложений (AADL)

Области применения DSL

- Описание разметки данных (HTML, CSS)
- Описание интерфейсов приложений (XAML, XUL, QML)
- Описание моделей данных (DDL, ER)
- Описание доступа к данным в СУБД (SQL, DAX)
- Описание доступа к структурам данных (XQuery, XPath)
- Описания сетевых протоколов (SDL)
- Описания форматов данных (ASN.1)
- Описания грамматик языков (EBNF, flex, bison)
- Описания манипуляций над данными (DML)
- Описание процессов (BPMN, Дракон, ПОСП)

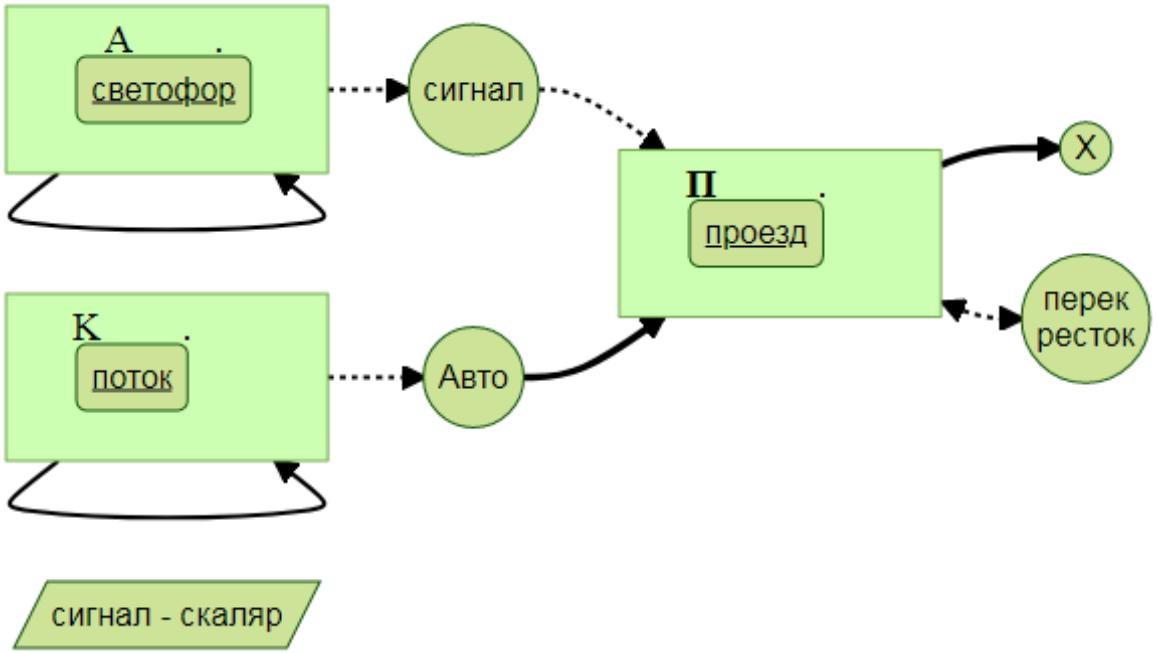
Языки описания графов / DOT, GraphML /

```
graph graphname {
        // label - видимое название вершины
        a [label="Foo"];
        // shape - определение формы вершины
        b [shape=box];
        // color - определение цвета ребра
        a -- b -- c [color=blue];
        // style - определение стиля ребра
        b -- d [style=dotted];
}
```

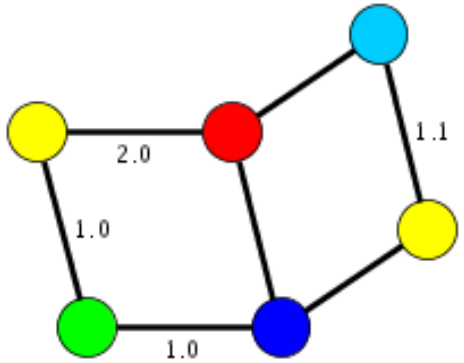


Язык описания диаграмм / mermaid.js/

```
flowchart LR
    svet -.-> a2((сигнал))
    a2 -.-> xsect
    k1((Авто)) ==> xsect
    src1 -.-> k1
    xsect ==> ax((X))
    xsect <-.-> k2((перек<br/>ресток))
    subgraph src1
     ["К    "]
     c1(<u>поток</u>)
    end
    subgraph xsect ["<b>П</b>  "]
     b(<u>проезд</u>)
    end
    subgraph svet ["А        "]
     a("<u>светофор</u>")
    end
   src1 --> src1
   svet --> svet
   h0[/сигнал - скаляр/]

  classDef blocks width:160px,font-size:20px,font-family:serif,text-align:left;
  class xsect,src1,svet blocks;
```



Язык описания графов /GraphML/




```
<?xml version="1.0" encoding="UTF-8"?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns
http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd">
<key id="d0" for="node" attr.name="color" attr.type="string">
<default>yellow</default>
</key>
<key id="d1" for="edge" attr.name="weight" attr.type="double"/>
<graph id="G" edgedefault="undirected">
<node id="n0">
<data key="d0">green</data>
</node>
<node id="n1"/>
<node id="n2">
<data key="d0">blue</data>
</node>
```

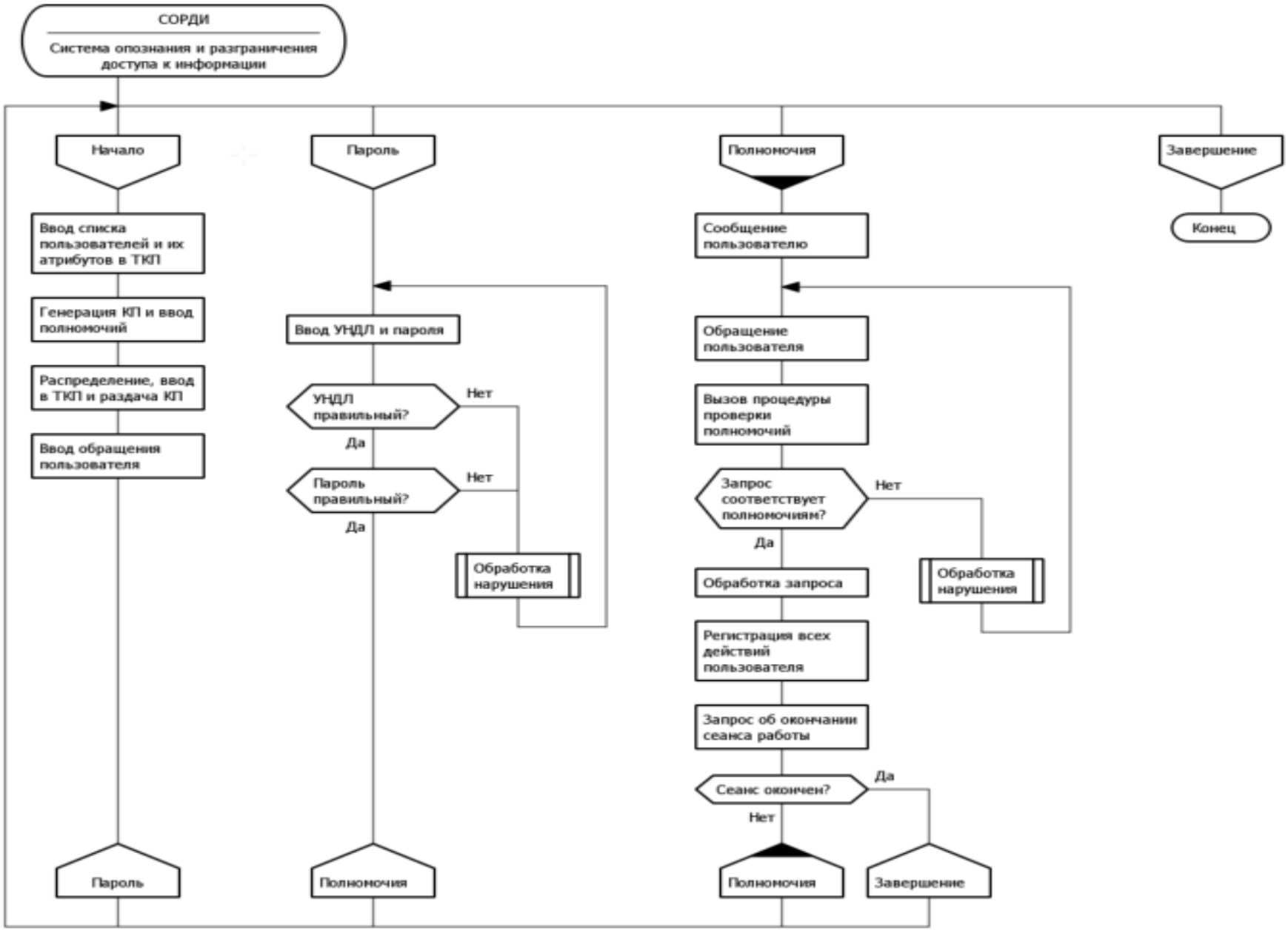
```
...
<edge id="e0" source="n0" target="n2">
<data key="d1">1.0</data>
</edge>
...
</graph>
```

Язык описания алгоритмов / ДРАКОН /

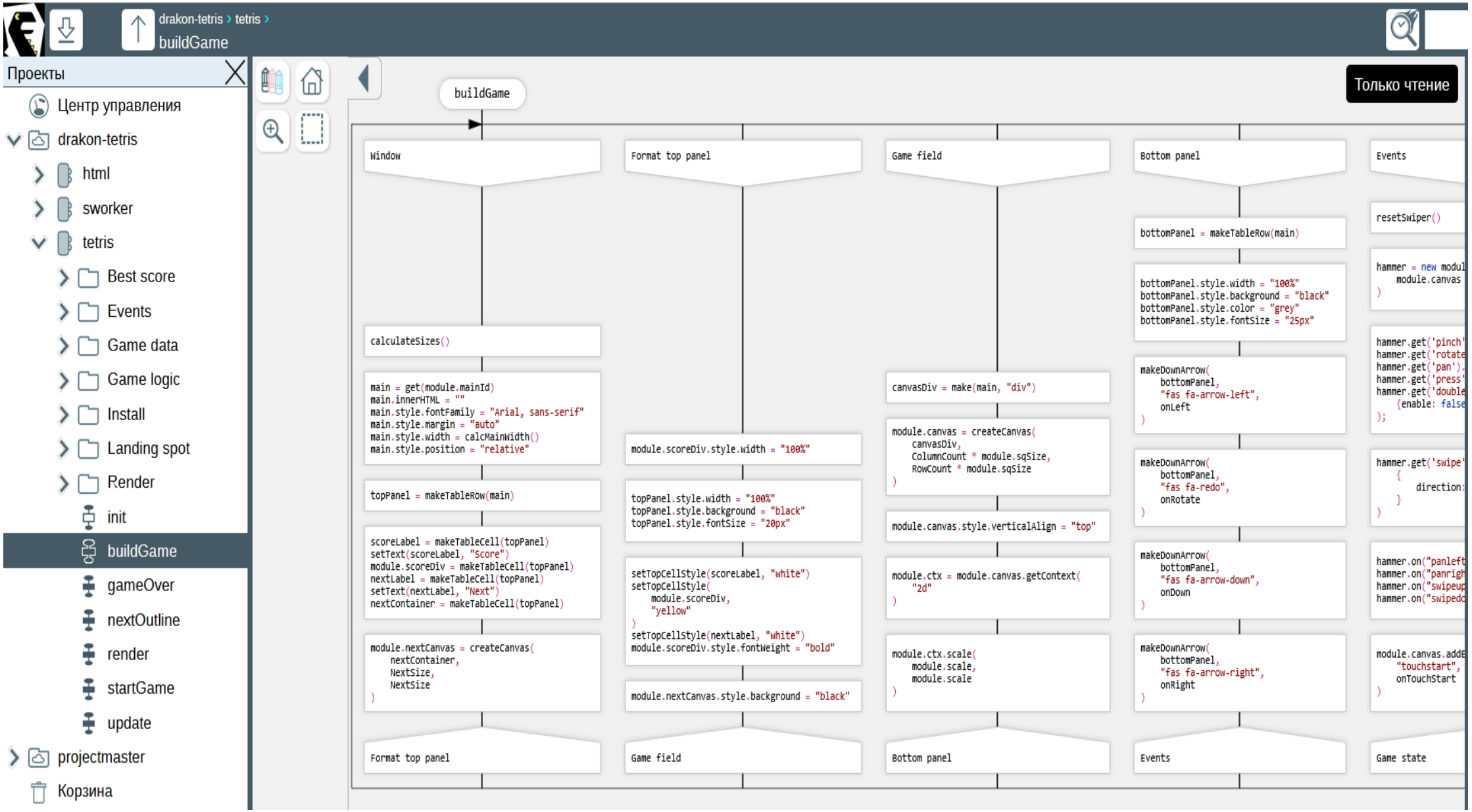
ДРАКОН — это графический алгоритмический язык,
созданный в ходе проекта космической системы Буран.
Инженеры создавали ДРАКОН, чтобы иметь дело с реальной
сложностью систем управления космическими аппаратами.
Авторы языка ДРАКОН спроектировали язык таким образом,
чтобы снизить когнитивные усилия при работе с алгоритмами.
Каждое правило и каждая функциональная возможность
ДРАКОНа помогает людям быстрее понимать программы.

Drakon.Tech перенёс язык ДРАКОН в веб-среду, позволяя разработчикам
применять эту технологию для веб-приложений.
Процесс разработки в Drakon.Tech включает следующие шаги:
- Разработчик рисует Дракон-схемы в Drakon.Tech IDE.
- Генератор кода создаёт JavaScript-функцию для каждой Дракон-схемы
- Drakon.Tech помещает эти функции в JS-файл, который можно выполнить
в браузере или Node.js.

Язык описания алгоритмов / ДРАКОН /



Язык алгоритмов ДРАКОН https://drakon.tech/



Последователи GPSS

JGPSS Framework
is discrete time simulation general-purpose open source tool written in Java

```
* Modelo de una cafeteria
* Atiende clients

clientes Generate   Uniform,15,8,0.0,0.0,0.0,0  ;<<--comment-->>
camarero Seize      camarero       ;<<--comment-->>
         Advance    Uniform,4,1   ;<<--comment-->>
         Release    camarero     ;<<--comment-->>
salida   Terminate  1           ;<<--comment-->>
```

Последователи GPSS

go-gpss

is a framework based on concept GPSS (General Purpose Simulation System)

https://godoc.org/github.com/soldatov-s/go-gpss

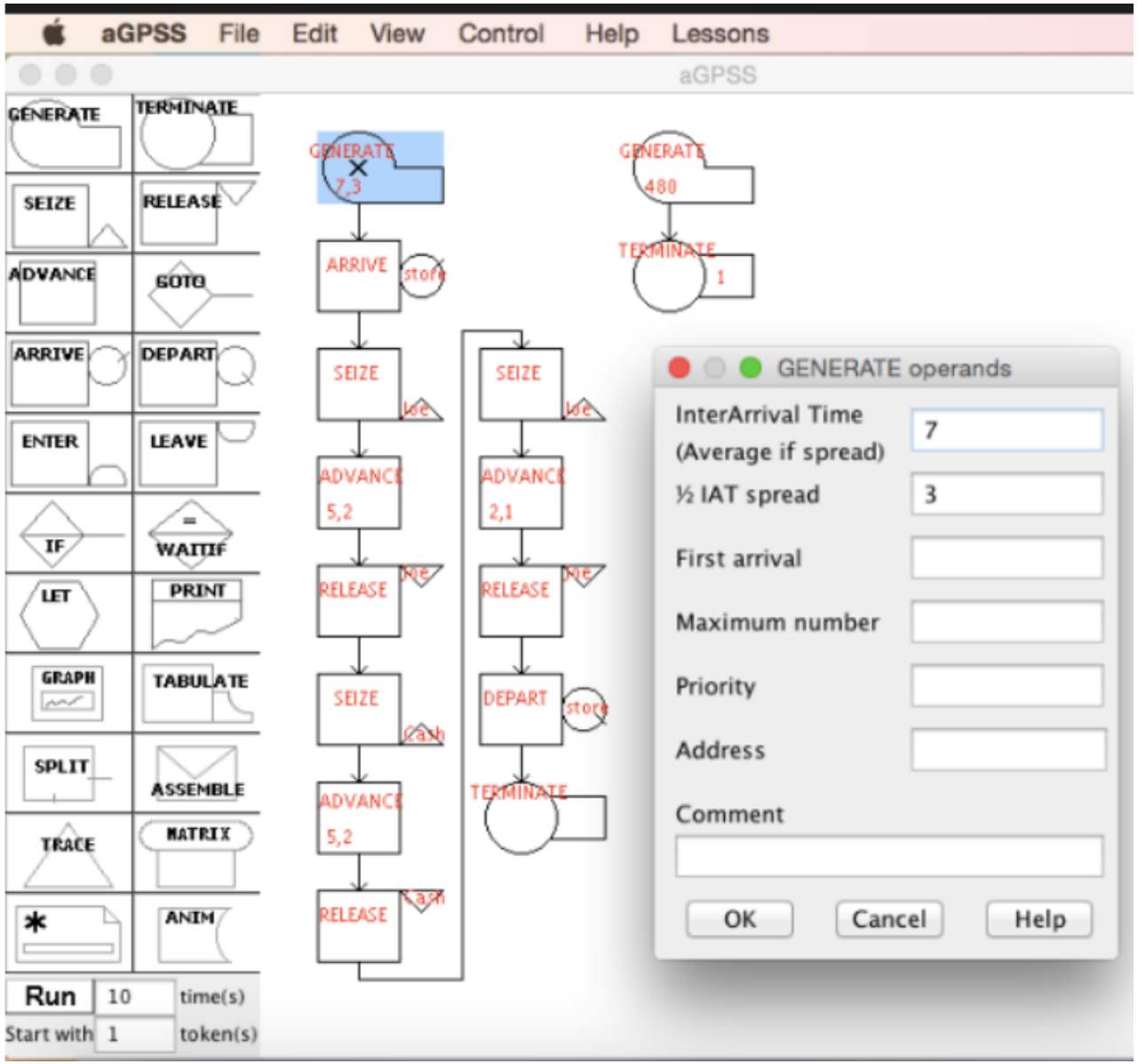
```Golang
// Build pipeline
// Generator -> Queue -> Facility -> Hole

p := objects.NewPipeline("Barbershop")
        AddObject(objects.NewGenerator("Clients", 18, 6, 0, 0, nil))
        AddObject(objects.NewQueue("Chairs"))
        AddObject(objects.NewFacility("Master", 16, 4))
        AddObject(objects.NewHole("Out"))
// Start simulation
p.Start(480)
<-p.Done
p.Report()
```

Последователи GPSS

aGPSS

DISCRETE EVENT SIMULATION ON THE MACINTOSH FOR BUSINESS STUDENTS

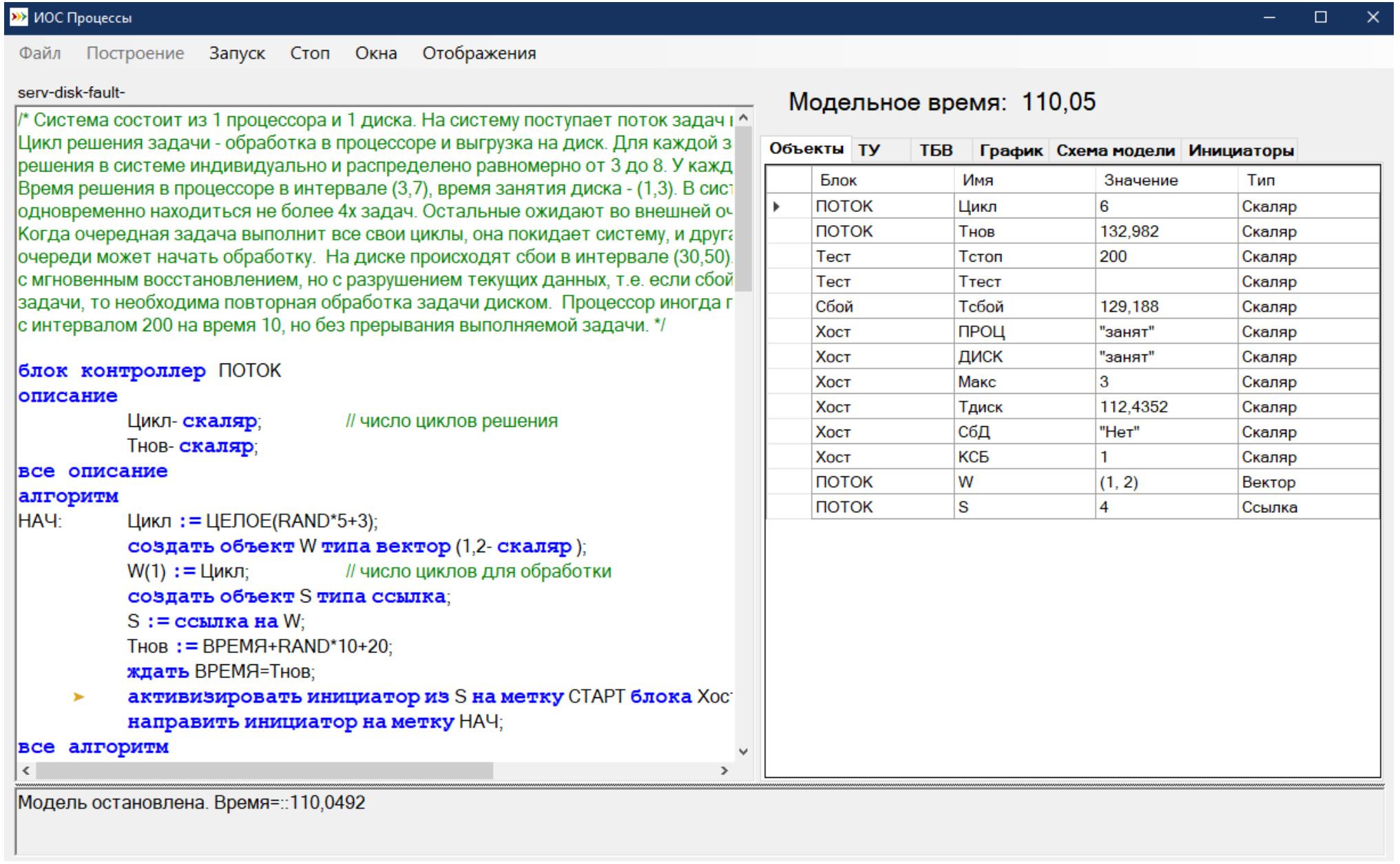
Prof. Ingolf Ståhl
Department of Economic Statistics
Stockholm School of Economics



Моделирование на основе ПОСП

ИОС «Процессы»

Проект разработки на C#

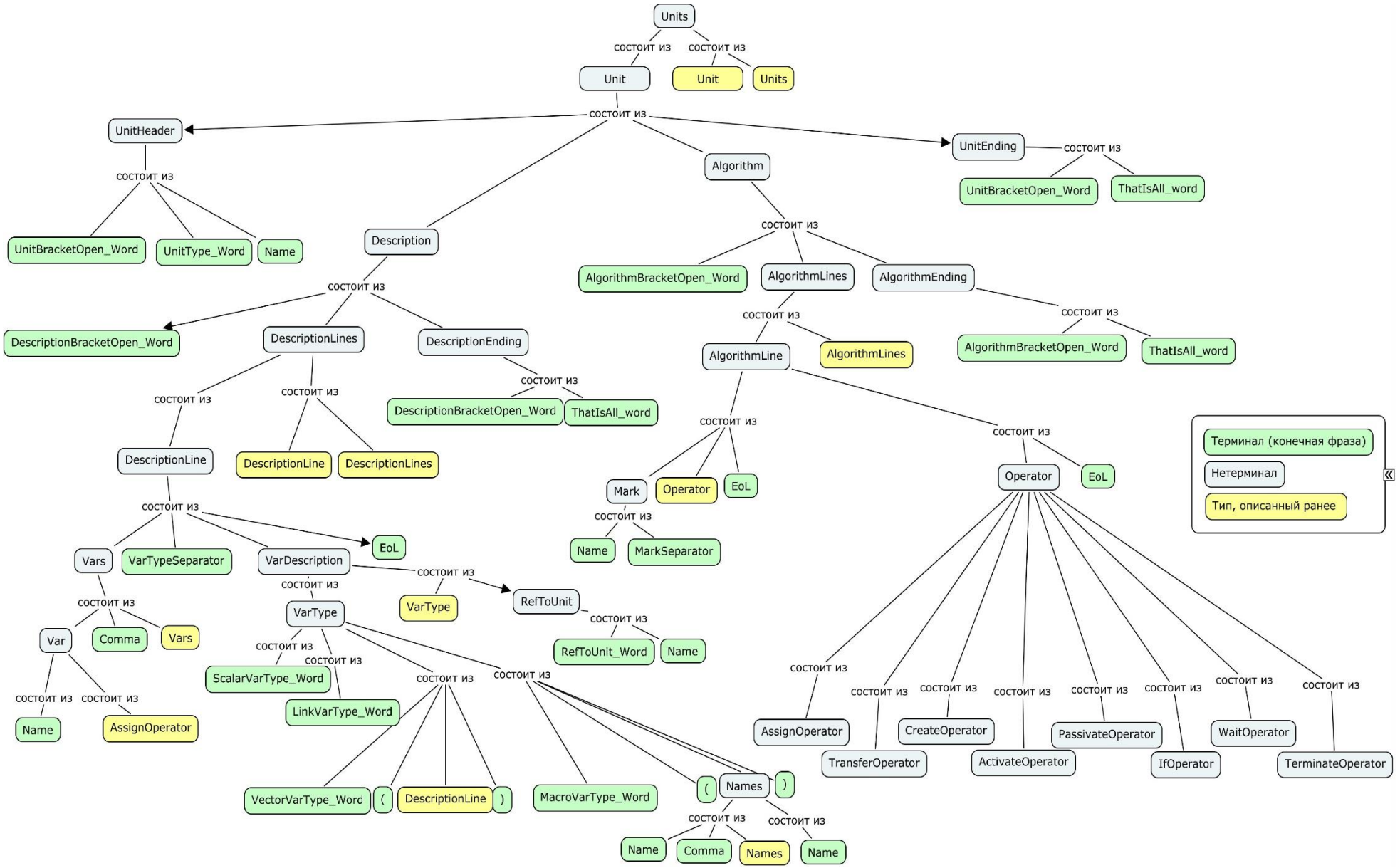


Моделирование на основе ПОСП

ИОС «Процессы»

Проект разработки на C#

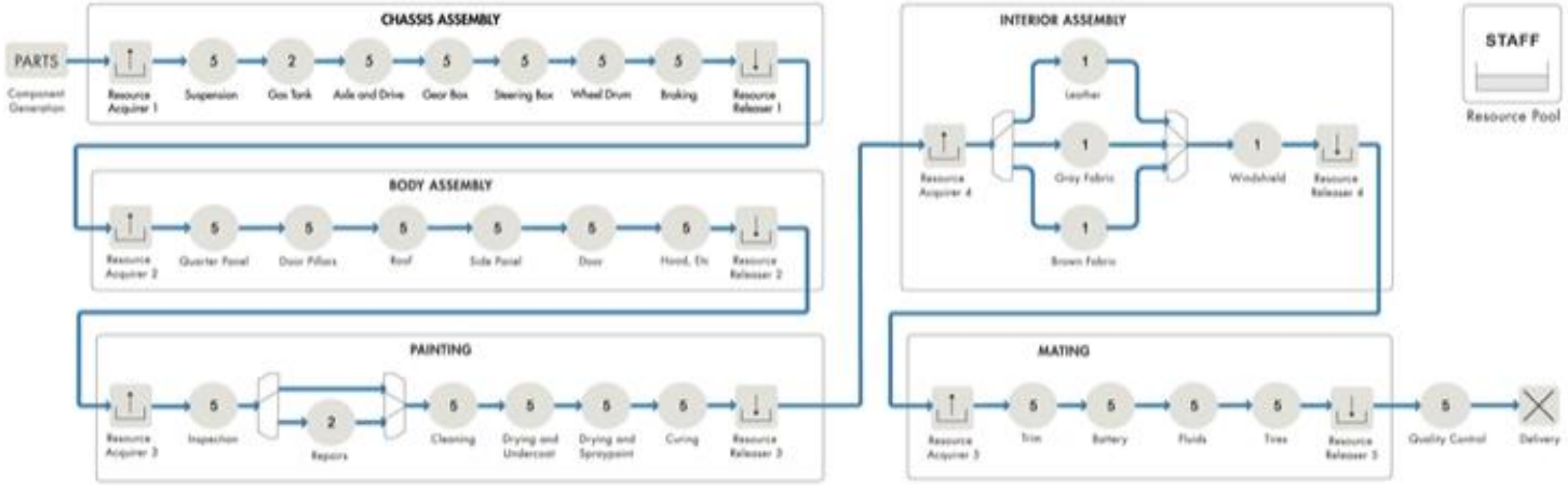
Схема синтаксиса ПОСП



Последователи GPSS

MathWorks MatLab

Stateflow предоставляет графический язык, который включает диаграммы перехода состояний, блок-схемы, таблицы перехода состояний и таблицы истинности.



Последователи GPSS

MathWorks MatLab

Stateflow предоставляет графический язык, который включает диаграммы перехода состояний, блок-схемы, таблицы перехода состояний и таблицы истинности.

Можно использовать **Stateflow**, чтобы описать, как алгоритмы MATLAB и модели Simulink реагируют на события и временные условия.

С помощью **Stateflow** можно проектировать и разрабатывать диспетчерский контроль, планирование задач, управление отказами, протоколы связи, пользовательские интерфейсы и гибридные системы.

SimEvents предоставляет механизм моделирования дискретных событий и библиотеку компонентов для анализа моделей систем, управляемых событиями, и оптимизации характеристик производительности (задержка, пропускная способность и потеря пакетов).

С помощью **SimEvents** можно изучать влияние времени выполнения задач и использования ресурсов на производительность вашей системы, проводить оперативные исследования для принятия решений, связанных с прогнозированием, планированием мощностей или управлением цепочкой поставок.

Методология создания DSL

Анализ: 1) определение спектра задач предметной области;

2) сбор информации о предметной области;

3) объединение знаний в семантические нотации и операции над ними;

4) проектирование DSL для лаконичного описания предметной области.

Реализация: 1) конструирование библиотеки семантических нотаций;

2) проектирование и реализация транслятора (транспилятор, transpiler) DSL-программ

Использование: 1) написание DSL программ для возможных приложений;

2) компиляция программ.

Транспайлеры используются во многих областях программирования.

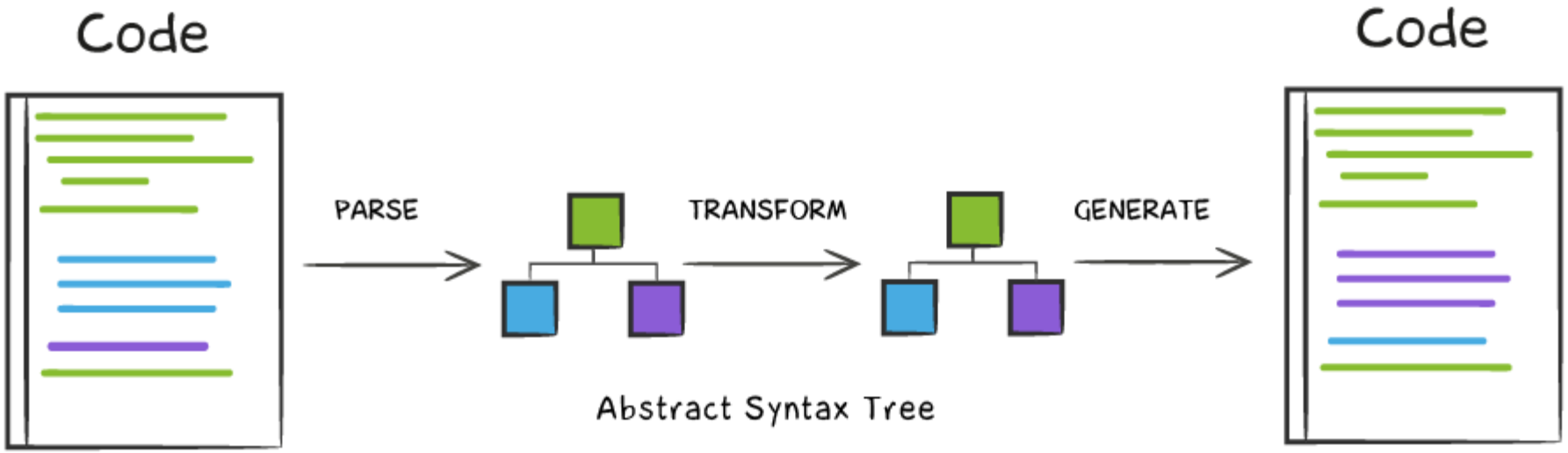
Можно выделить две основных области применения:

- Транспиляция одной версии языка в другую.

Поскольку языки программирования постоянно обновляются, возникают ситуации, когда разработчик уже пишет на новой версии языка, а среда, где используется код, поддерживает и работает только с предыдущими версиями. Например:
- для переходов между версиями стандарта ECMAScript используется транспайлер Babel;
- для переходов с версий Python 2.x на Python 3 можно использовать транспайлер 2to3;

- Транспиляция между разными языками.

Например, преобразование TypeScript в JavaScript, или среда Haxe.org



Исходный код преобразуется в абстрактное синтаксическое дерево (AST), далее AST трансформируется в структуру, которая соотносится с конечным кодом (AST), и из этой структуры генерируется необходимый новый код

Внешние DSL : шаги разработки

- Определение цели создания языка описания предметной области
- Выбор средств представления языка описания предметной области
- Разработка элементов языка описания предметной области
- Реализация языка описания предметной области инструментальными средствами

Инструментарий разработки DSL

/ Описание грамматик языков /

- ✓ расширенная форма Бэкуса-Наура (EBNF)
- ✓ JetBrains MPS (meta programming system)
- ✓ Lex + YACC
- ✓ JavaCC grammar
- ✓ ANTLR (ANother Tool for Language Recognition)
- ✓ Microsoft Visual Studio Modeling SDK (DSL Toolkit)

Описание грамматик языков в форме Бэкуса-Наура (EBNF)

<цифра> ::= 0-9

<число> ::= {<цифра>}+

<вырж> ::= <терм>[(\+|\-)<вырж>]

<терм> ::= <множ>[(*|/)<терм>]

<множ> ::= [<множ>^]<степ>

<степ> ::= \(<вырж>\)|<идентификатор>|<число>

<программа> ⇒ < вырж >

⇒ < вырж >

⇒ < идентификатор >=< множ >

⇒ a =< множ >

⇒ a =< терм > + < терм >

⇒ a =< идентификатор > + < терм >

⇒ a = b+ < терм >

⇒ a = b+ < константа >

Формальная система определения синтаксиса, в которой одни синтаксические категории последовательно определяются через другие, используется для описания контекстно–свободных формальных грамматик. Предложен Никлаусом Виртом, и является расширенной переработкой форм Бэкуса-Наура, отличается более «ёмкими» конструкциями, позволяющими при той же выразительной способности упростить и сократить описание.

https://dic.academic.ru/dic.nsf/ruwiki/493710

Основные преимущества DSL

- ✓ DSL позволяет выражать решения в форме идиом и на высоком уровне абстракций
- ✓ DSL программы лаконичны, самодокументированы для больших объемов, могут быть повторно использованы
- ✓ Использование DSL увеличивает продуктивность, надежность, удобство сопровождения и переносимость
- ✓ DSL заключают в себе предметные знания, что делает возможным их сохранение и повторное использование
- ✓ DSL разрешают валидацию данных и оптимизацию на уровне предметной области
- ✓ Использование DSL улучшает тестируемость программ

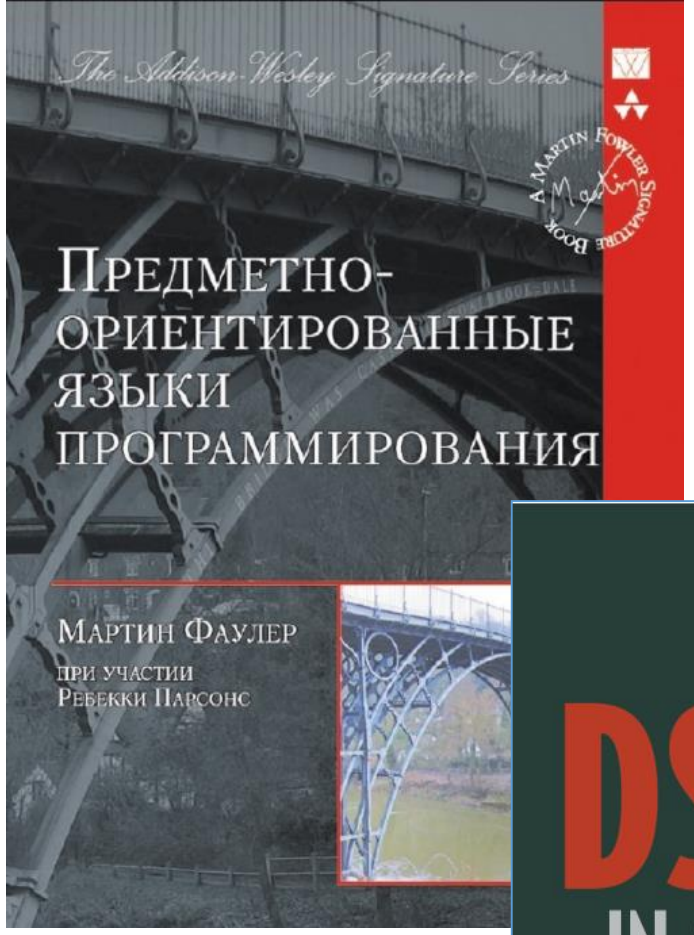
ИСТОЧНИКИ

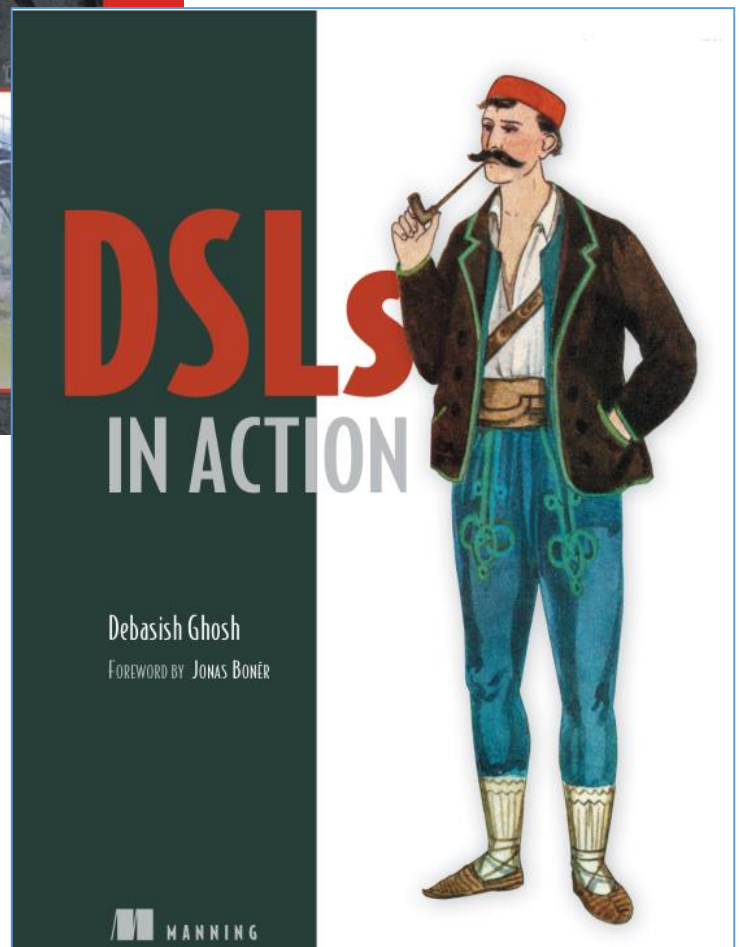
http://martinfowler.com

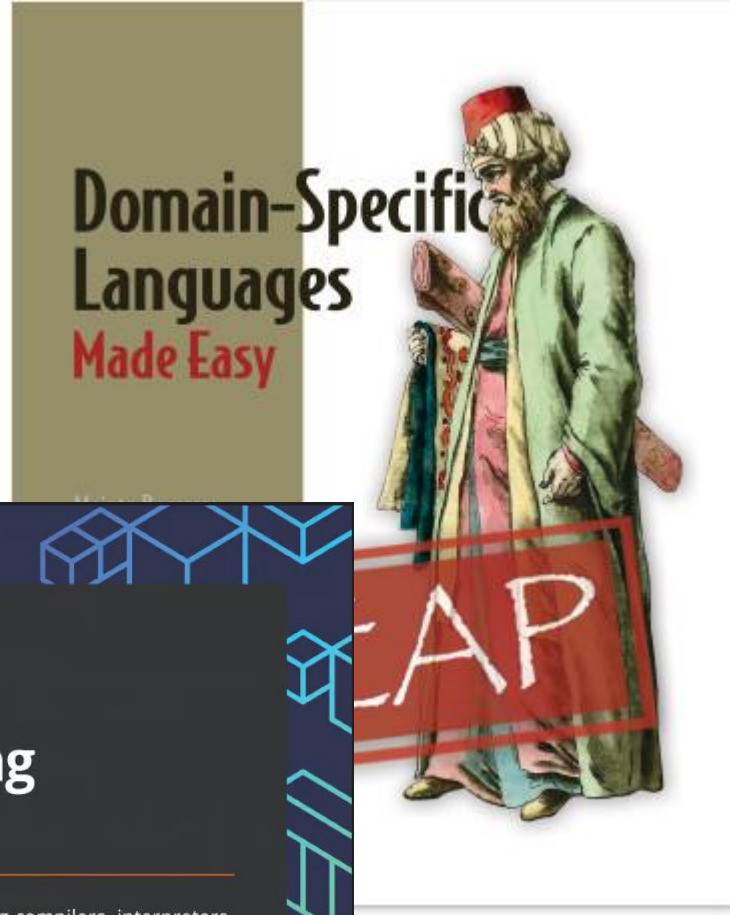
Jetbrains Meta Programming System

https://www.jetbrains.com/ru-ru/mps/concepts/domain-specific-languages/

https://ru.wikipedia.org/wiki/Предметно-ориентированный_язык


The Addison-Wesley Signature Series
A Martin Fowler Signature Book
Предметно-ориентированные языки программирования
Мартин Фаулер
при участии
Ребекки Парсонс


DSLs
IN ACTION
Debasish Ghosh
Foreword by Jonas Bonér
MANNING


Domain-Specific
Languages
Made Easy

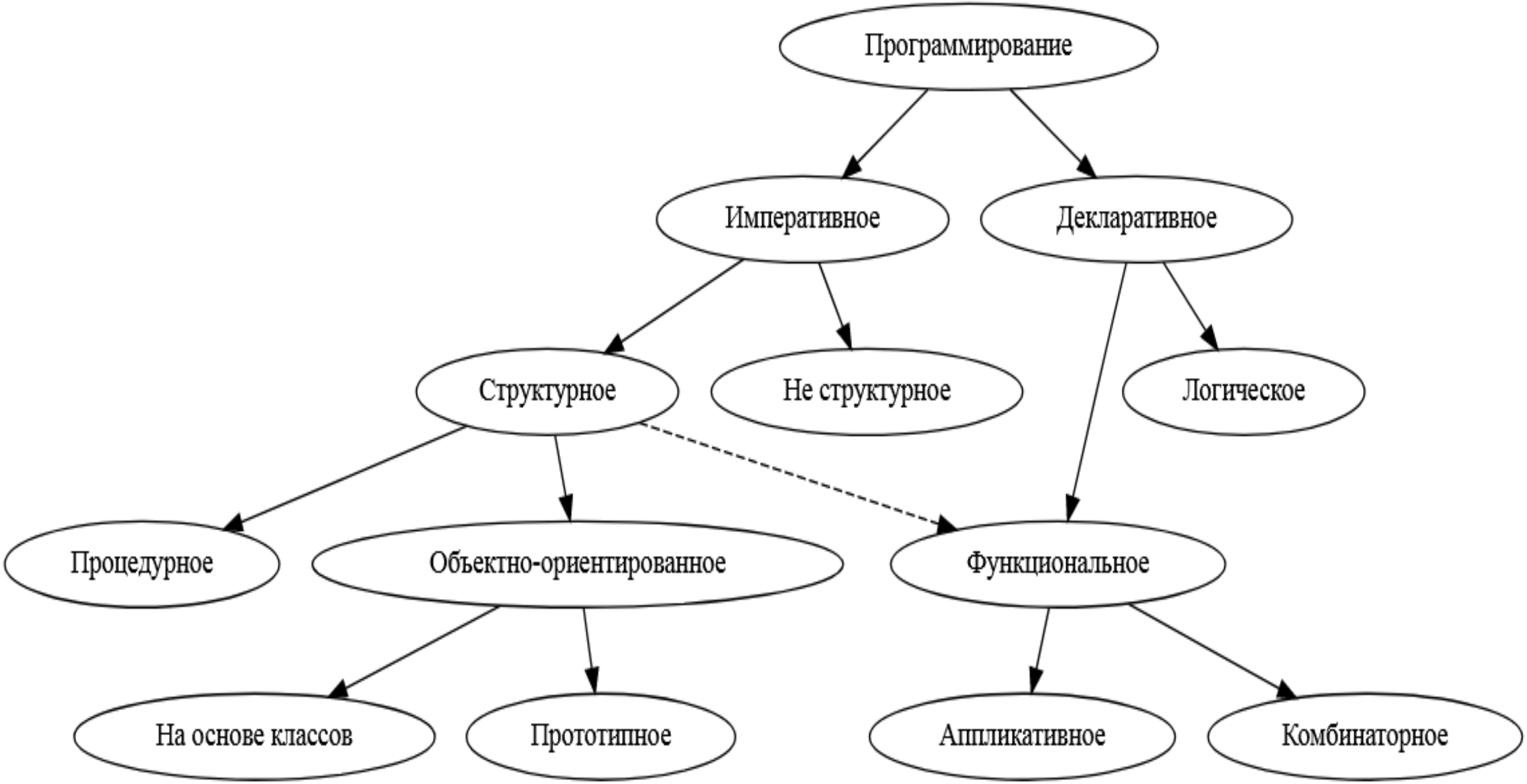

Build Your Own
Programming
Language
A programmer's guide to designing compilers, interpreters, and DSLs for solving modern computing problems
Clinton L. Jeffery

Один из главных аспектов интеллектуальной мощи людей – не просто язык, а многоязычие. Чем больше языков знает индивидуум, тем выше его интеллект, тем многогранней его сознание, тем масштабней и сложней его разум.

✓ Язык – это не только самый совершенный способ передачи информации при коммуникациях, но и «свет разума»:
- это бесценный инструмент для организации, обработки и структурирования информации;
- это особый механизм восприятия и познания мира, в том числе, самих себя.

✓ Знания многих специализированных языков (математики, химии, музыки, поэзии, живописи, кино, языков программирования, создания промптов для ИИ и пр.) - дает колоссальное преимущество их носителю в познании мира, в общении с ним и (что выделяет людей из всей живой природы) в актах творчества (создания в мире нового, ранее не существовавшего) и саморазвития способностей творца.

✓ Естественные языки …выполняют похожую роль со специализированными языками. Они формируют для человека уникальный взгляд на сложности и многообразие мира и дают ему уникальный инструмент его познания.


Программирование
Императивное
Декларативное
Структурное
Не структурное
Логическое
Процедурное
Объектно-ориентированное
Функциональное
На основе классов
Прототипное
Аппликативное
Комбинаторное