Новиков Ф.А.

Методы повышения качества алгоритмизации предметных областей на основе определения проблемно-ориентированных языков

Диссертация на соискание ученой степени доктора технических наук по специальности 05.13.11 Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей

2011

Структура презентации

- Введение (Общая характеристика работы) 3 – 12
  - Актуальность темы; экспериментальная база
- Глава 1 (Технология прикладного программирования) 13 – 20
  - Назначение и область применения методов алгоритмизации предметных областей
- Глава 2 (Алгоритмизация предметных областей) 21 – 29
  - Методы алгоритмизации, основанные на программной интерпретации понятий предметной области
- Глава 3 (Автоматный метод определения DSL) 30 – 55
  - Метод определения предметно-ориентированных языков, основанный на императивных моделях описания поведения
- Глава 4 (Структурный синтез программ) 56 – 72
  - Методы автоматического синтеза программ, основанные на декларативных языках спецификации
- Заключение (Выводы) 73 – 80
  - Внедрение; положения, выносимые на защиту; избранные публикации по теме диссертации

Общая характеристика работы

- Неформально:
  - Длительное время (с 1974 года по настоящее время)
  - Во многих случаях (не менее 17)
  - Автор лично участвовал (играя разные роли)
  - В алгоритмизации предметных областей (нескольких)
  - И сделал полезные выводы из наблюдений (4 вывода)
- Формально:
  - Актуальность темы и цели работы
  - Практическая значимость и внедрение результатов
  - Содержание диссертации по главам
  - Научная новизна и положения, выносимые на защиту

Актуальность темы (1 из 4)

- Повышение качества алгоритмизации предметных областей актуально
  - Предметная область :≈ структуры данных + процедуры обработки + типовые задачи
    - Бухгалтерия = (счёт +…) + (проводка +…) + (начисление з/п +…)
    - Астрономия = (орбита +…)+ (интегрирование +…) +(эфемериды +…)
  - Алгоритмизация := методы решения типовых задач + программное обеспечение + технология применения
    - 1С = (главная книга +…) + (СУБД+…) + (Конфигурация +…)
    - ЭРА = Небесная механика + ППП + язык СЛОН
  - Качество := доля решаемых типовых задач = степень удовлетворенности сильного пользователя

Актуальность темы (2 из 4)

- Методы алгоритмизации различны и улучшаемы
  - Математическая модель
    - Пакет прикладных программ (библиотека)
      - Язык программирования общего назначения
  - Описательная модель предметной области
    - Domain Specific Language (DSL)
      - Доверительный программный фонд
  - DSL :=
    - предметно-ориентированный язык :=
      - ориентирован на структуры данных (T_EX)
    - проблемно-ориентированный язык :=
      - ориентирован на процедуры обработки (Excel)
    - пользователе-ориентированный язык :=
      - ориентирован на решение типовых задач (VBA)

Актуальность темы (3 из 4)

- Определение и реализация DSL дает значительный выигрыш в качестве алгоритмизации
  - Значительный выигрыш :≈ кодирование ↓ + ошибки ↓ + выразительность ↑ + повторное использование ↑ + … (Ward)
    - T_EX ↔ (Word+MathType): время ввода формул = 5 : 1
    - SQL ↔ (DBF+Clipper): число ошибок = 1 : 3
  - Классификация методов определения и реализации DSL
    - Наивно-бессознательные методы (пионерские)
      - Использование GUI, специальная терминология
    - Языково-ориентированные методы (классические)
      - Грамматические описания, визуальное программирование
    - Модельно-ориентированные методы (модные)
      - Модель предметной области (МПО)

Актуальность темы (4 из 4) = Цели работы

- Исследование способов построения, изложения, теоретического обоснования, практического применения и сравнения различных методов алгоритмизации предметных областей на основе определения DSL:
  - Влияния DSL на процесс разработки прикладного ПО = ?
  - Настройка DSL на конкретную предметную область = ?
  - Абстрактный синтаксис DSL (система понятий языка) = ?
  - Конкретный синтаксис DSL (внешний вид) = ?
  - Операционная семантика DSL (как работает) = ?
- Создание новых методов определения DSL, повышающих качество алгоритмизации предметных областей

История вопроса = экспериментальная база = практическая значимость (1 из 4)

Название	Год	С кем	Что	#
FP/FPG (Fortran Preprocessor / Fortran Program Generator)	1979	Бабаев И.О.	язык для описания расширения фортрана проблемно-ориентирован-ными типами данных (компьютерная алгебра)	1
STEREOL (STEpwise REfinement Oriented Language)	1979		язык для составления программ методом пошагового уточнения (программирование)	1
Декарт (DESCARTES — DESCribe your Area, Realize the Target and Extract the Solution)	1980 1986	Бабаев И.О. Лавров С.С. Петрушина Т.И.	язык описания МПО, БД и ППП, ориентированный на автоматический синтез программ (система СПОРА) вычислительные задачи	7

История вопроса = экспериментальная база = практическая значимость (2 из 4)

Название	Год	С кем	Что	#
СЛОН (СЛежение и Обработка Наблюдений)	1986 1991	Красинский Скрипниченко	язык для решения задач эфемеридной астрономии, основанный на табличном подходе к обработке данных	5
Improvement	1990 1992	Иванова Т.В. Парийская Е.Ю.	входной язык пакета приклад-ных программ для улучшения параметров математических моделей из астрономических наблюдений методом наименьших квадратов	3
AstroTOP (Astro Table Oriented Programming)	1991 1997	Крашенинников Назаров А.А. Скрипниченко	предметно-ориентируемый язык, основанный на таблич-ном подходе к обработке данных	9

История вопроса = экспериментальная база = практическая значимость (3 из 4)

Название	Год	С кем	Что	#
GUIDL	1995 2004	Яценко А.Д.	язык публикаций, позволяющий описывать алгоритмы работы пользователя графического интерфейса	6
SVITA	1998 1999	Нецветаева Г.А. Парийская Е.Ю	входной язык системы верстки табличных изданий	3
DELTA (DELphi + TAble)	2006 2010	Михеева В.Д. Скрипниченко	язык для решения задач эфемеридной астрономии, основанный на табличном подходе к обработке данных и встроенный в язык Паскаль как включающий язык	1
DiaDel (DIAgram DEfinition Language)	2007	Степанян К.Б.	язык описания графических языков, использующих нотацию графоподобных диаграмм	2

История вопроса = экспериментальная база = практическая значимость (4 из 4)

Название	Год	С кем	Что	#
AutoLanD	2008 2010	Тихонова У.Н.	семейство визуальных языков определения проблемно-ориентированных языков на конечных автоматах	6
aCMK (автоматизи-рованная Система Менеджмента Качества)	2008	Клебан В.О.	язык описания бизнес-процессов и документооборота на конечных автоматах	1
ИПС (Исполняемые Программные Спецификации)	2009 2010	Новосельцев В.Б.	язык спецификаций, основанный на автоматическом синтезе вычислительных программ	2
AMPLE 3 (Adaptable Minor PLanets Ephemerides)	2010	Нецветаева и другие	входной язык заданий для ППП в области динамики малых тел Солнечной системы	1

Содержание работы: Введение

- Проблемы разработки прикладного программного обеспечения
- Необходимость решения этих проблем для достижения целей алгоритмизации в конкретных предметных областях
- Понятие качества алгоритмизации как меры достижения целей алгоритмизации
- Эффективность применения предметно-ориентируемых и проблемно-ориентированных языков и методов при алгоритмизации
- Важность опоры на формальные методы
- Тенденция визуального программирования
- Постановка задачи: **на основе исследования известных методов определения DSL разработать новые методы определения DSL, повышающие качество алгоритмизации предметных областей по сравнению с известными методами**.

Глава 1. Назначение и область применения методов алгоритмизации предметных областей

- **Задачи главы**: введение в проблемы разработки прикладного программного обеспечения, обзор существующих решений этих проблем и изложение защищаемого автором подхода к их решению.
- **План главы**:
  - ***1.1. Технология разработки прикладного программного обеспечения***
    - Эпиграф: программисты неуправляемы, программы ненадежны, а программирование – рискованный бизнес
  - ***1.2. Средства и методы алгоритмизации предметных областей***
    - Эпиграф: невозможно автоматизировать хаос
  - ***1.3. Технология алгоритмизации предметной области***
    - Эпиграф: все программы уже написаны, остается их найти, исправить ошибки и научиться использовать

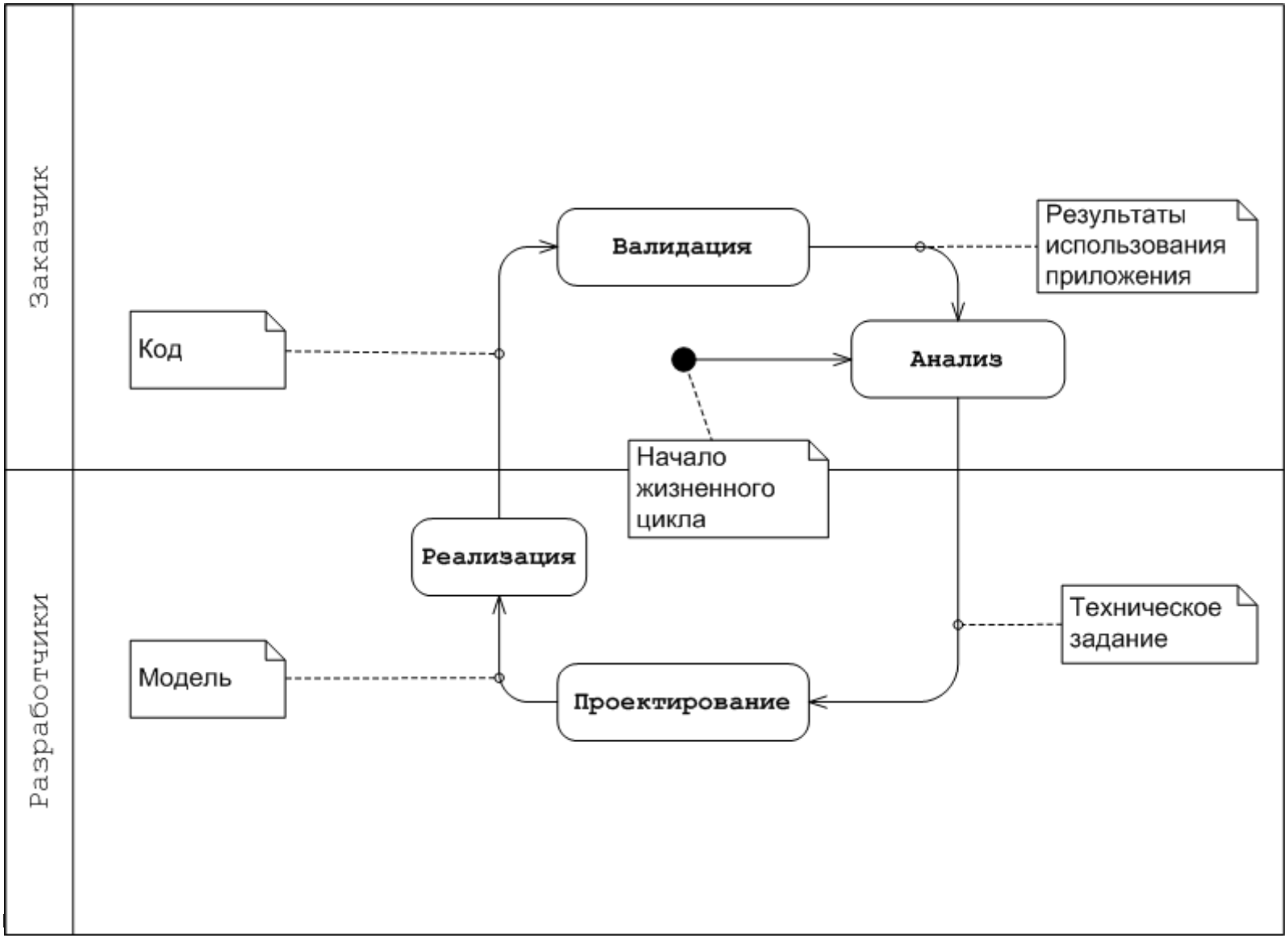
Технология программирования (1 из 2)

- Предмет технологии программирования
  - Программирование — процесс создания программистом программы (информационной структуры) [, предназначенной для последующего исполнения (компьютером)].
  - Технология программирования — это совокупность методов и средств, позволяющая наладить производственный процесс программирования.
- Противоречия программирования
  - Статика (программа) ↔ Динамика (выполнение)
  - Качество (произвол в комбинациях) ↔ Количество ($7 \leftrightarrow 10^9$)
- Типы программных проектов
  - Системное программирование
  - Прикладное программирование
  - Модельное программирование

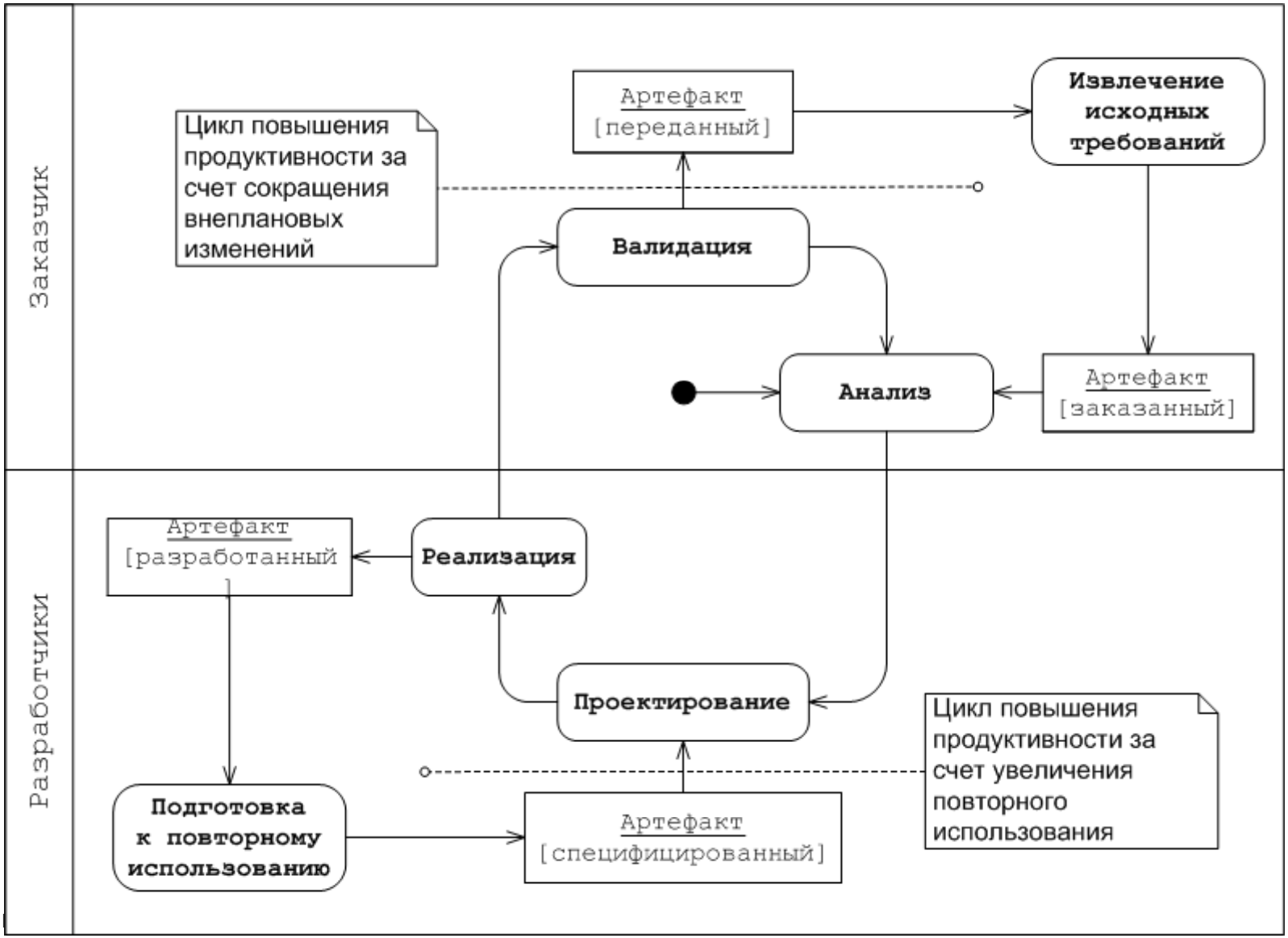
Технология программирования (2 из 2)

- Составные части технологии программирования
  - модель процесса – на уровне организации
  - модель команды – на уровне проекта
  - дисциплина программирования – на уровне программиста
- Задачи технологии программирования
  - Снижение совокупной стоимости владения
  - Повышение надежности
  - Повышение продуктивности разработки
- Методы повышения продуктивности
  - сокращение объема внеплановых изменений артефактов
  - увеличение объема повторно использованных артефактов
  - ускорение движения информации за счет использования DSL для представления всех артефактов

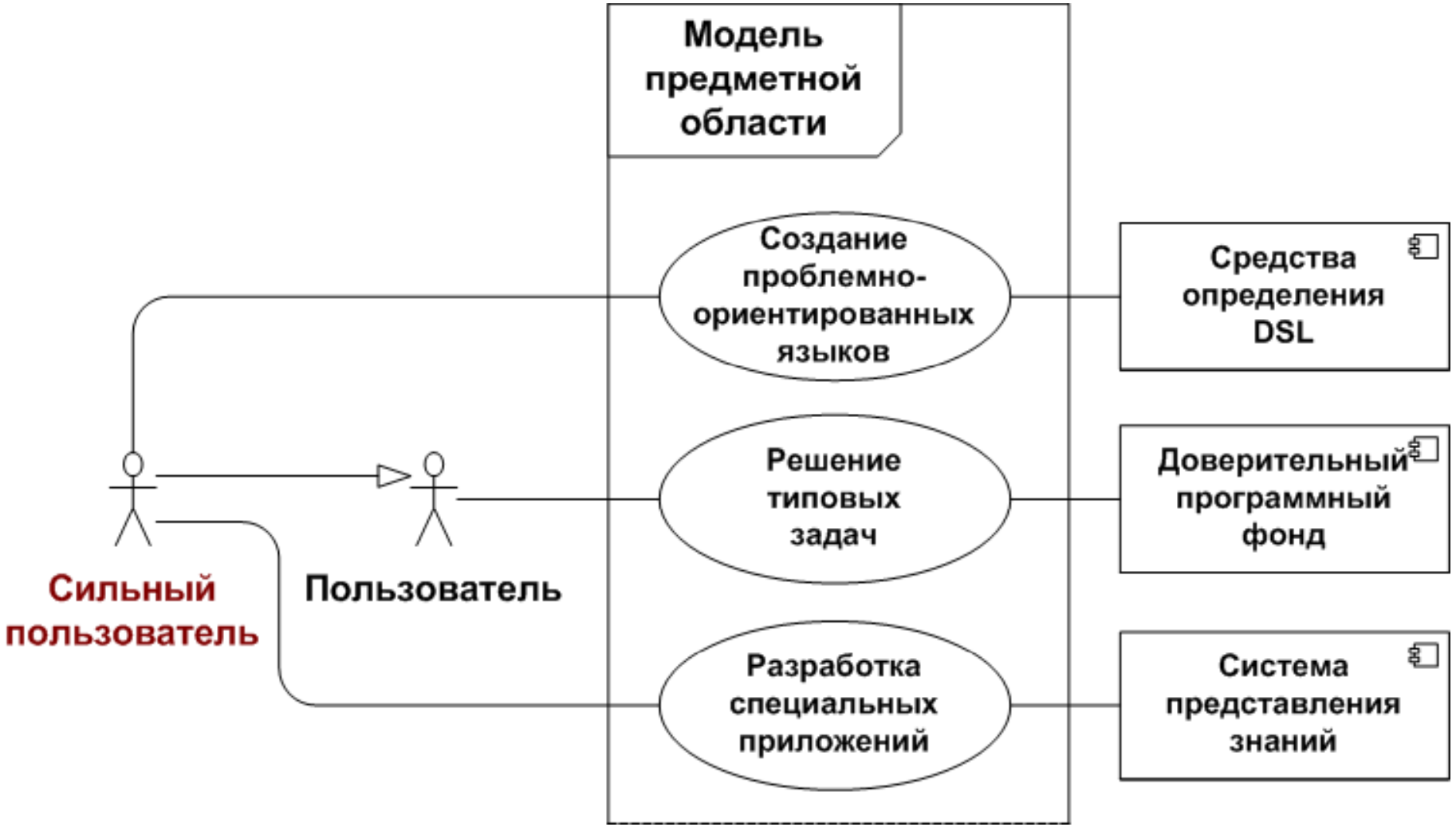
Итеративный процесс разработки ПО



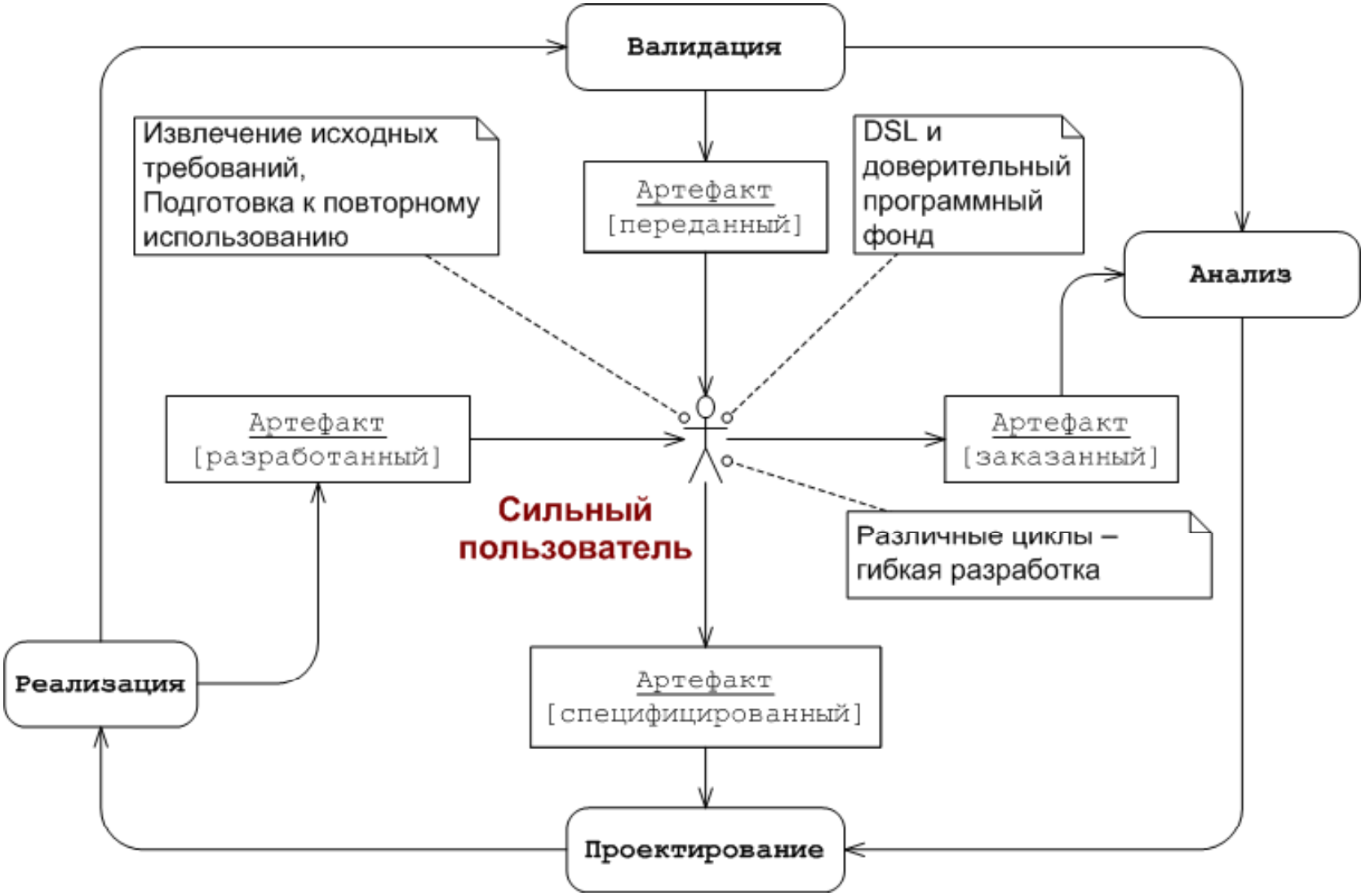
Циклы повышения продуктивности



Модель использования модели предметной области



Сильный пользователь
≈ эксперт в предметной области



Выводы по главе 1

- Языково-ориентированное программирование является одним из основных средств повышения продуктивности разработки прикладного программного обеспечения
- Предметно- и проблемно-ориентированные языки предполагают построение моделей предметных областей
- Модели предметных областей опираются на готовый (доверительный) программный фонд
- Сильный пользователь является важным фактором повышения качества алгоритмизации предметной области
- **Концепция циклов повышения продуктивности** и объяснение влияния проблемно-ориентированных языков на разработку прикладного программного обеспечения является первым выносимым на защиту результатом

Глава 2. Методы алгоритмизации, основанные на программной интерпретации понятий предметной области

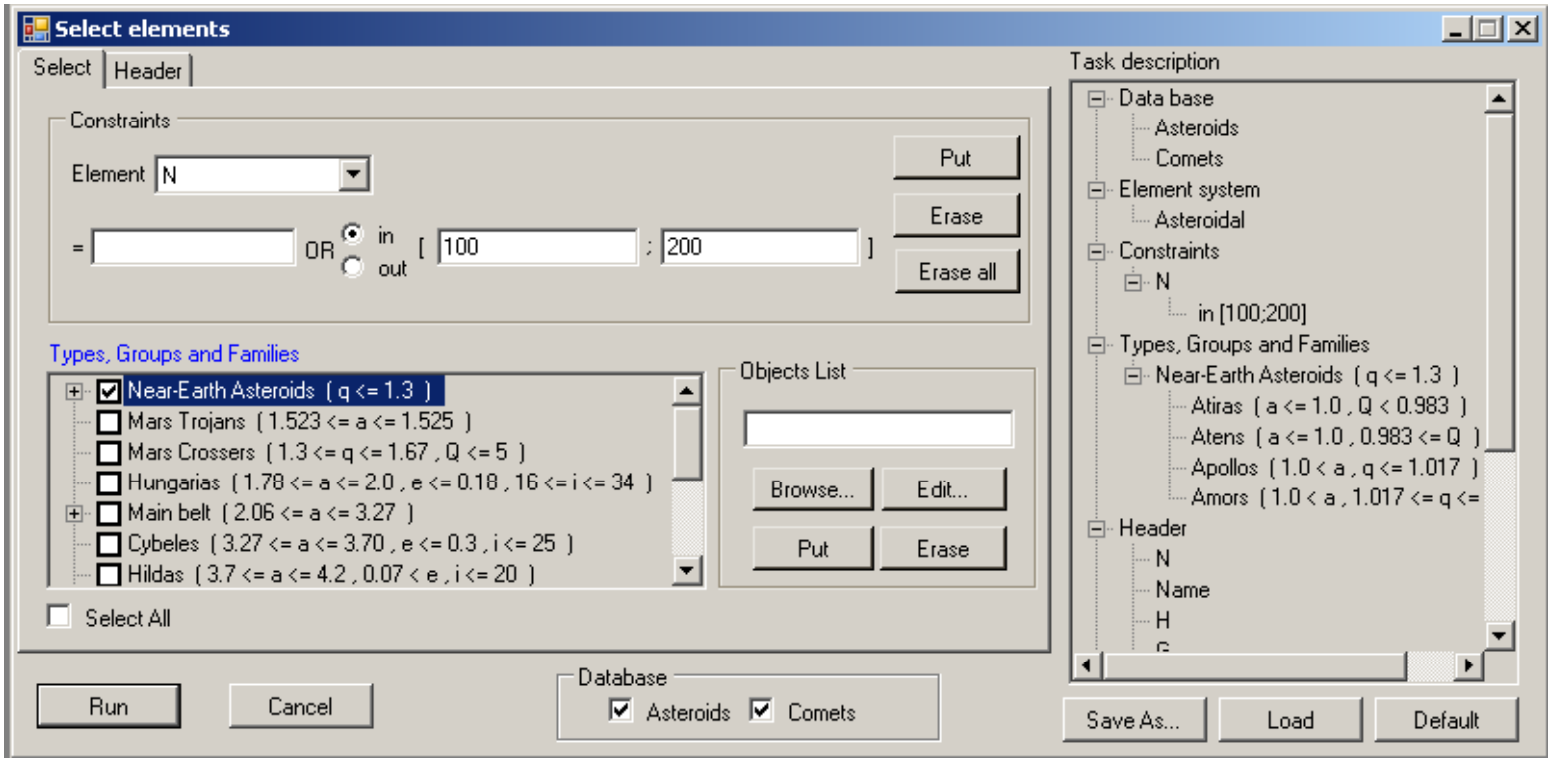
- **Задача главы**: демонстрация примеров проблемно-ориентированных языков, разработанных диссертантом традиционными методами, анализ результатов их применения и накопление материала для последующего обобщения
- **План главы**:
  - ***Параграф 2.1. Командные предметно-ориентированные языки***
    - Эпиграф: команды делятся на подготовительные и исполнительные.
  - ***Параграф 2.2. Визуальное конструирование программ***
    - Эпиграф: если программисты называют себя инженерами, то пусть покажут свои чертежи!
  - ***Параграф 2.3. Проблемно-ориентированные парадигмы***
    - Эпиграф: высшая степень экспертизы – наличие *алгоритма* решения типовых задач в данной предметной области.
- **Рассматриваются шесть примеров:**
  - **Ample 3, GUIDL, DiaDeL, aCMK, SVITA, СЛОН+ТОП+Дельта**
    - **Наблюдения: что влияет на качество алгоритмизации**

Классификация предметно-ориентированных языков

- **Командные предметно-ориентированные языки**
  - Развитый пакет прикладных программ или библиотека модулей
  - Состав и порядок действий, которые нужны для решения задачи, пользователь определяет сам по ходу дела и не сообщает системе заранее
    - Ничего не добавляют к качеству алгоритмизации
- **Графические императивные языки**
  - Пользователь описывает решаемую задачу в целом
  - Последовательность требуемых действий задается в виде графической схемы или диаграммы
  - Система осуществляет решение задачи, опираясь на доверительный программный фонд
    - Наглядность и автоматизация процесса выполнения
- **Декларативные проблемно-ориентированные парадигмы**
  - Существует алгоритм решения [типовых] задач предметной области, и этот алгоритм так или иначе внесен в систему
  - Пользователь только ставит задачу, а система сама строит программу ее решения, сама подбирает подходящие шаги и сама выполняет их
    - Наивысшее качество алгоритмизации за счет выразительной силы

1. AMPLE 3 – Adaptable Minor PLanets Ephemerides

- Оконный графический интерфейс пользователя = командный предметно-ориентированный язык
- Полезно иметь формат всех команд в явном виде, например, в формате XML



2. GUIDL – GUI Description Language

- Язык публикаций, предназначенный для чтения и исполнения человеком
- Командный язык для записи пошаговых процедур
- Псевдографика, значки, полиграфия → обратно к иероглифам

- Пример: изменение макета и заливка фона
  - Главная
  - Слайды
  - Изменение макета

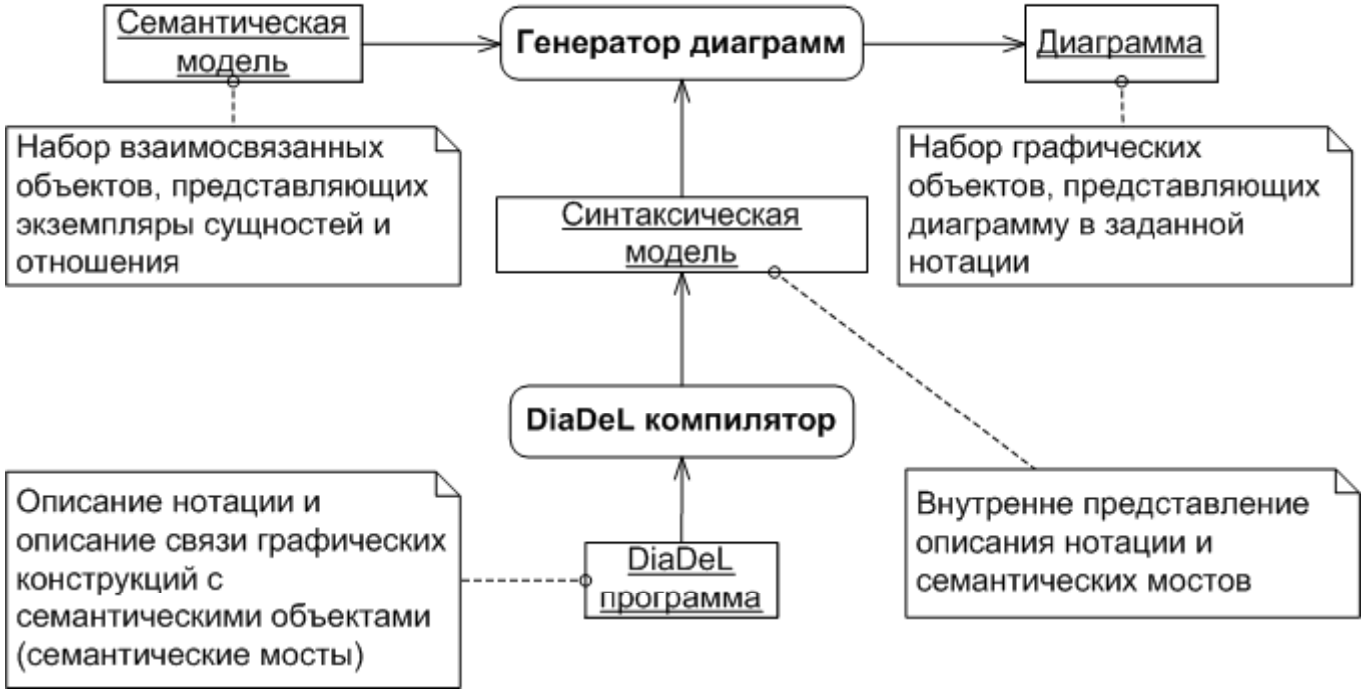
  // *Откроется палитра*

  1L Два объекта
  - Рисование
  - Заливка фигуры

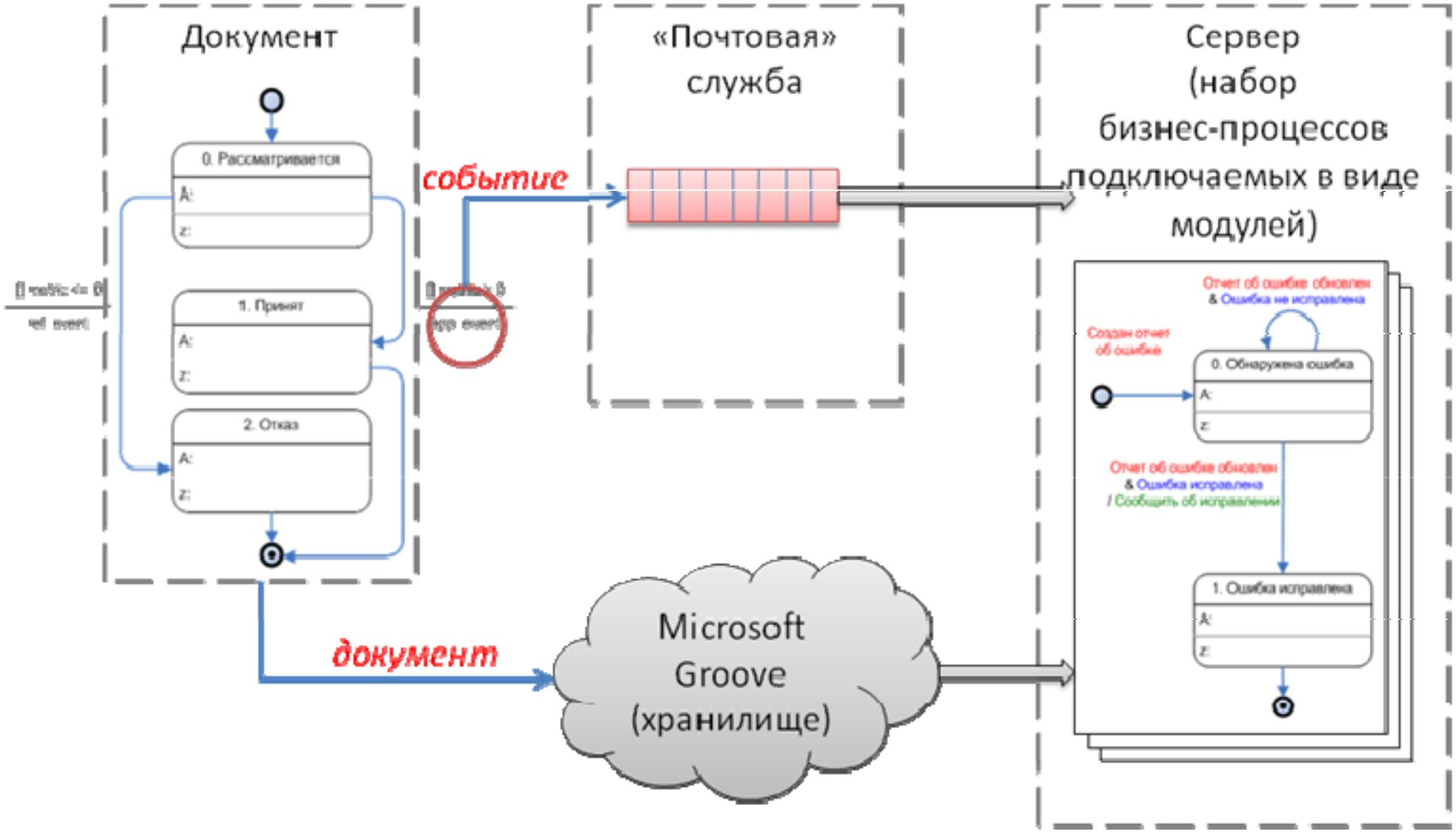
  1L Стандартные цвета[2]

3. DiaDeL – Diagram Definition Language

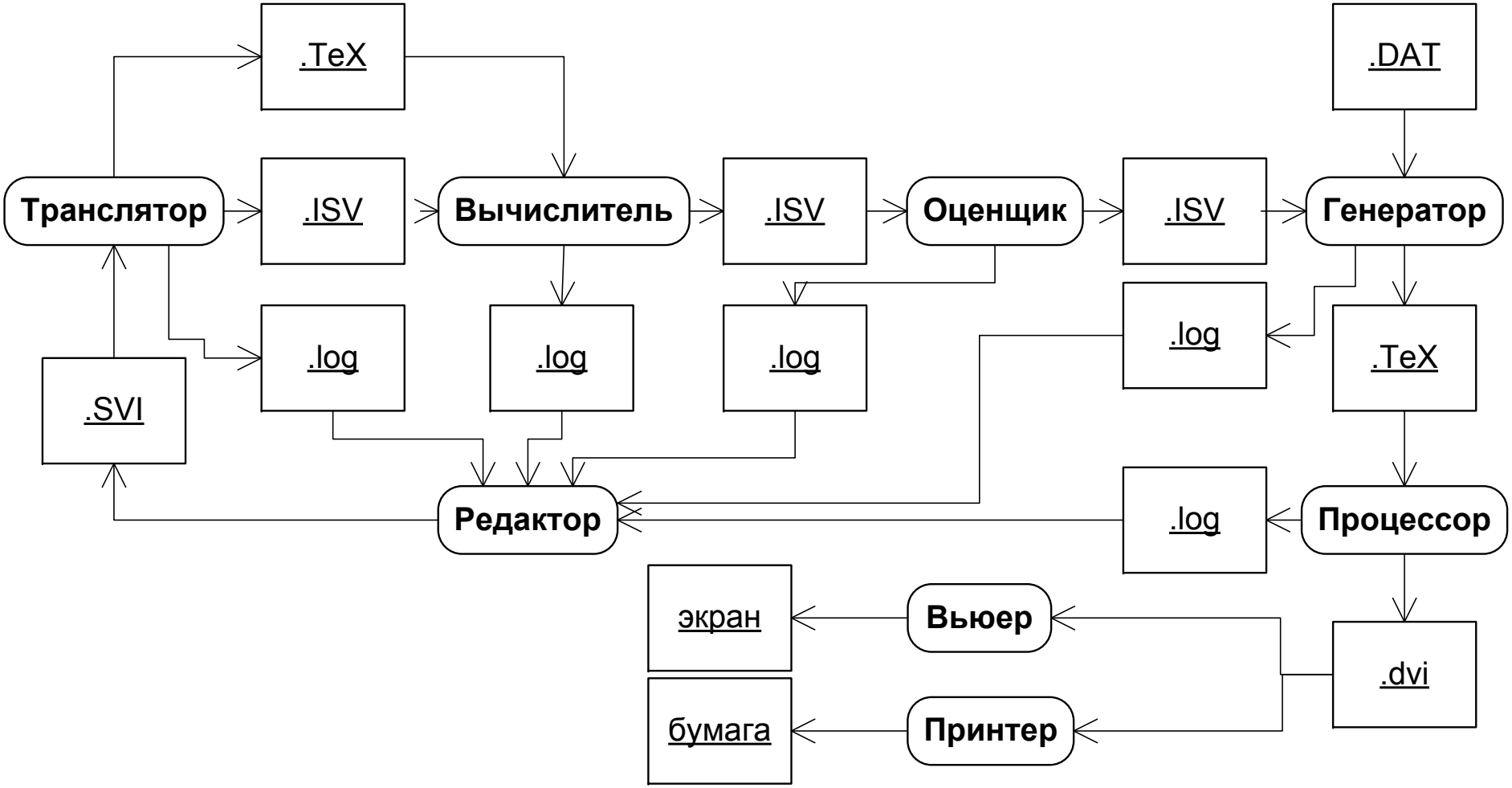
- Язык описания нотации графоподобных диаграмм
- Позволяет задать способ графического отображения любой заранее заданной системы объектов
- Имеет графические примитивы, соответствующие современному стилю рисования диаграмм



4. аСМК – автоматизированная СМК



5. СВИТА – Система Вёрстки ИТА



6. Парадигма таблично-ориентированного программирования

- Состояние предметной области – кортеж значений величин
- Двойственная природа таблиц: структура данных ↔ итератор

- Сложение. T = T1 + T2 = **for** E1 ∈ T1 **do yield** (E1 ∪ Default(E2–E1));
  for E2 ∈ T2 **do yield** (Default(E1–E2) ∪ E2)
  т.е. сложение таблиц — последовательное выполнение двух циклов
- Умножение.T = T1 * T2 = **for** E1 ∈ T1 **do for** E2 ∈ T2 **do yield** (E1∪ E2)
  т.е. умножение таблиц — вложенность циклов
- Итерация. T = n T1 = **for** i **from** 1 **to** n **do** T1
  т.е. итерация — цикл со счетчиком
- Проекция. T = T1 | s = **for** E1 ∈ T1 **do yield** ($E1|_s$)
  т.е. проекция — локализация переменных
- Выборка. T = T1 ? p = **for** E1 ∈ T1 **do if** p(E1) **then yield** (E1)
  т.е. выборка — условный оператор в теле цикла
- Действие. T = T1 f = **for** E1 ∈ T1 **do yield** (f(E1))
  т.е. действие — вызов процедуры в теле цикла

Выводы по главе 2

- Проблемно-ориентированные языки можно разделить на три класса:
  - командные проблемно-ориентированные языки,
  - графические императивные предметно-ориентированные языки
  - декларативные предметно-ориентированные языки
- Наиболее продуктивными являются декларативные языки
- Экспертное знание предметной области является непреложным условием достижения высокого качества алгоритмизации
  - Эксперт должен быть обеспечен адекватными программными средствами для построения модели предметной области
  - В прикладном программировании универсальные языки и «профессиональные» программисты менее продуктивны, чем проблемно-ориентированные языки и «сильные» пользователи.
- **Парадигма таблично-ориентированного программирования** и ее применение для алгоритмизации предметных областей, связанных с обработкой экспериментальных данных, является вторым выносимым на защиту результатом

Глава 3. Метод определения предметно-ориентированных языков, основанный на императивных моделях описания поведения

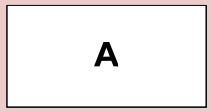
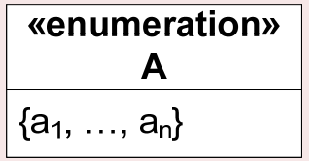
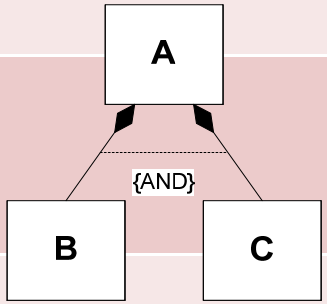
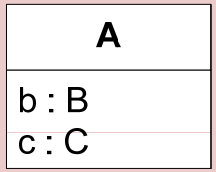
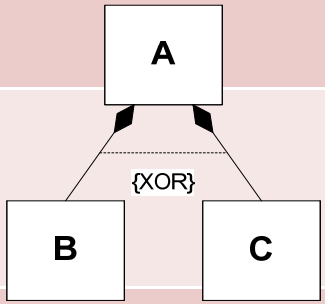
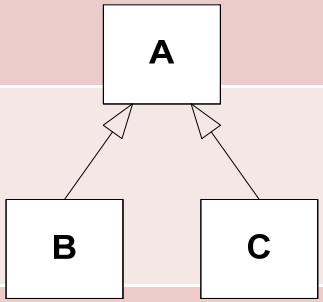
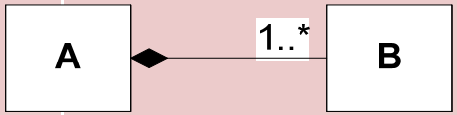
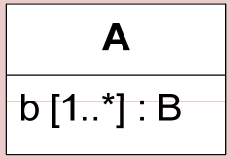
- **Задача главы**: представление оригинального нового метода определения проблемно-ориентированных языков на основе метамоделей и систем взаимодействующих автоматов.
- **План главы**:
  - ***Параграф 3.1. Автоматный метод определения языков***
    - Эпиграф: автоматное определение языка автоматически является реализацией языка!
  - ***Параграф 3.2. Парадигма автоматного программирования***
    - Эпиграф: область применения автоматного программирования безгранична.
  - ***Параграф 3.3. Конкурентные преимущества и открытые вопросы автоматного метода***
    - Эпиграф: всё познается в сравнении.

Шаги автоматного метода

1. **Определение абстрактного синтаксиса как иерархической композиции конструкций языка**
2. **Определение метамодели как абстрактного синтаксиса, дополненного системой неиерархических отношений между конструкциями языка**
3. **Определение конкретного синтаксиса как распознавателя, конструирующего абстрактную программу по её представлению**
4. **Определение операционной семантики как интерпретатора абстрактных программ**

Структура определения языка

Методика построения абстрактного синтаксиса по заданной грамматике

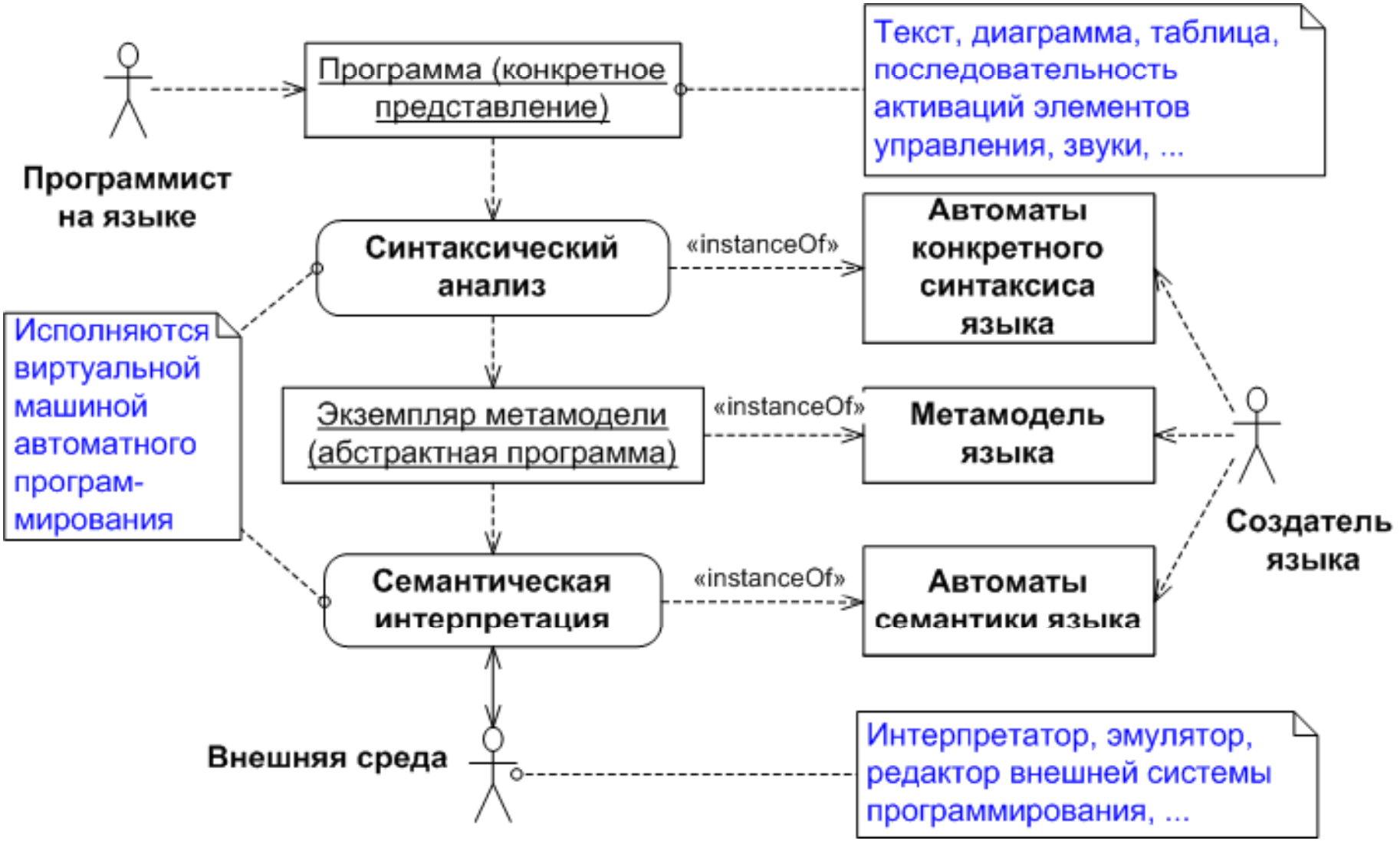
Грамматика	Выражение в метамодели	Сокращенная форма
α A ::=		
β A ::= a_1 | … | a_n		
γ A ::= B C		
δ A ::= B | C		
ε A ::= B | B A		

Пример = мини язык множеств

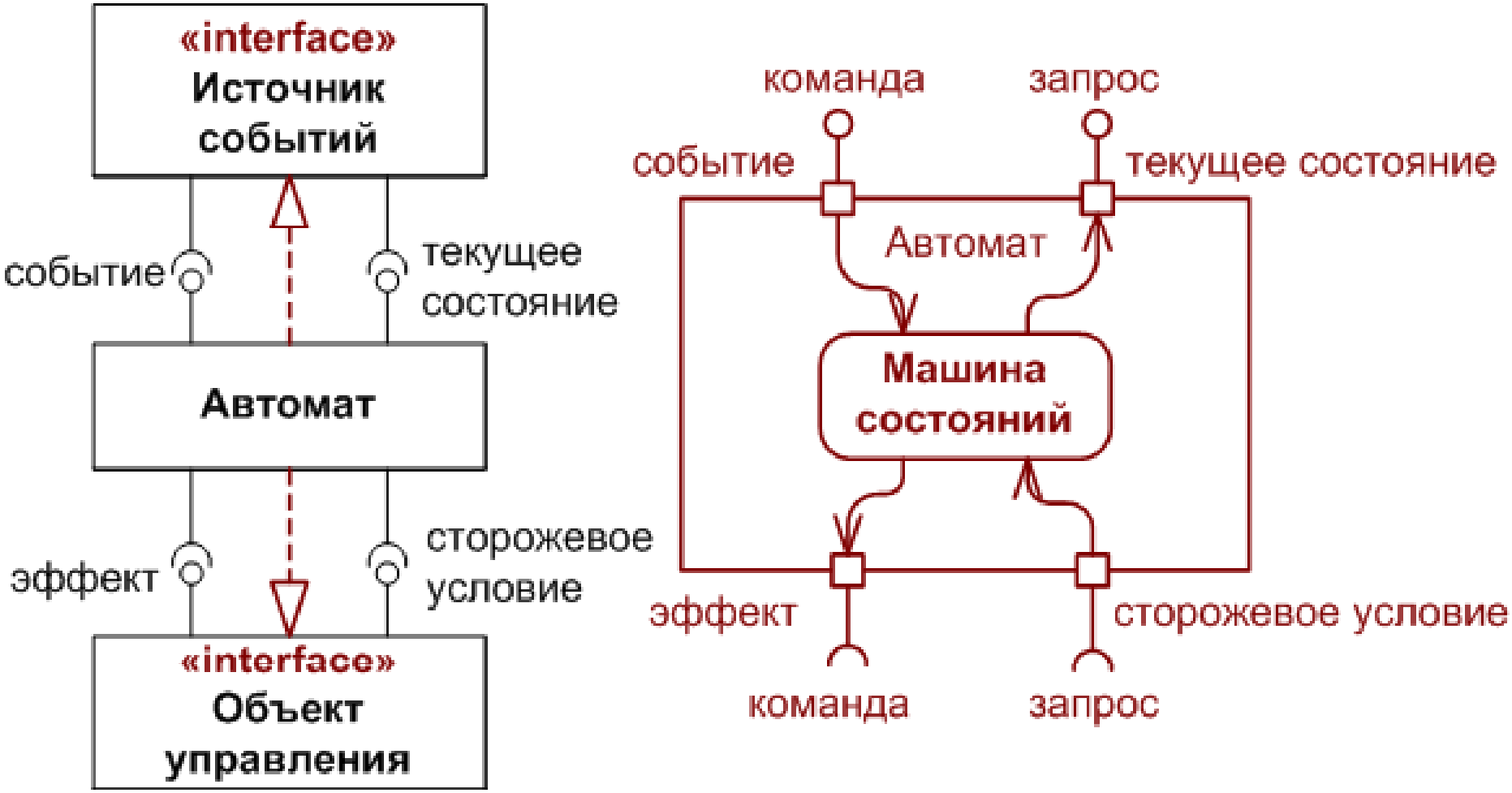
1. **Программа** ::= **Предложение** . | **Предложение** ; **Программа**
2. **Предложение** ::= **Имя** = **Выражение**
3. **Выражение** ::= **Выражение Операция Выражение** |
   Имя | { **Перечисление** }
4. **Операция** ::= ∪ | ∩
5. **Имя** ::= A | B | …| X | Y | Z
6. **Перечисление** ::= **Буква** | **Буква** , **Перечисление**
7. **Буква** ::= a | b | … | x | y | z

Метамодель мини языка множеств

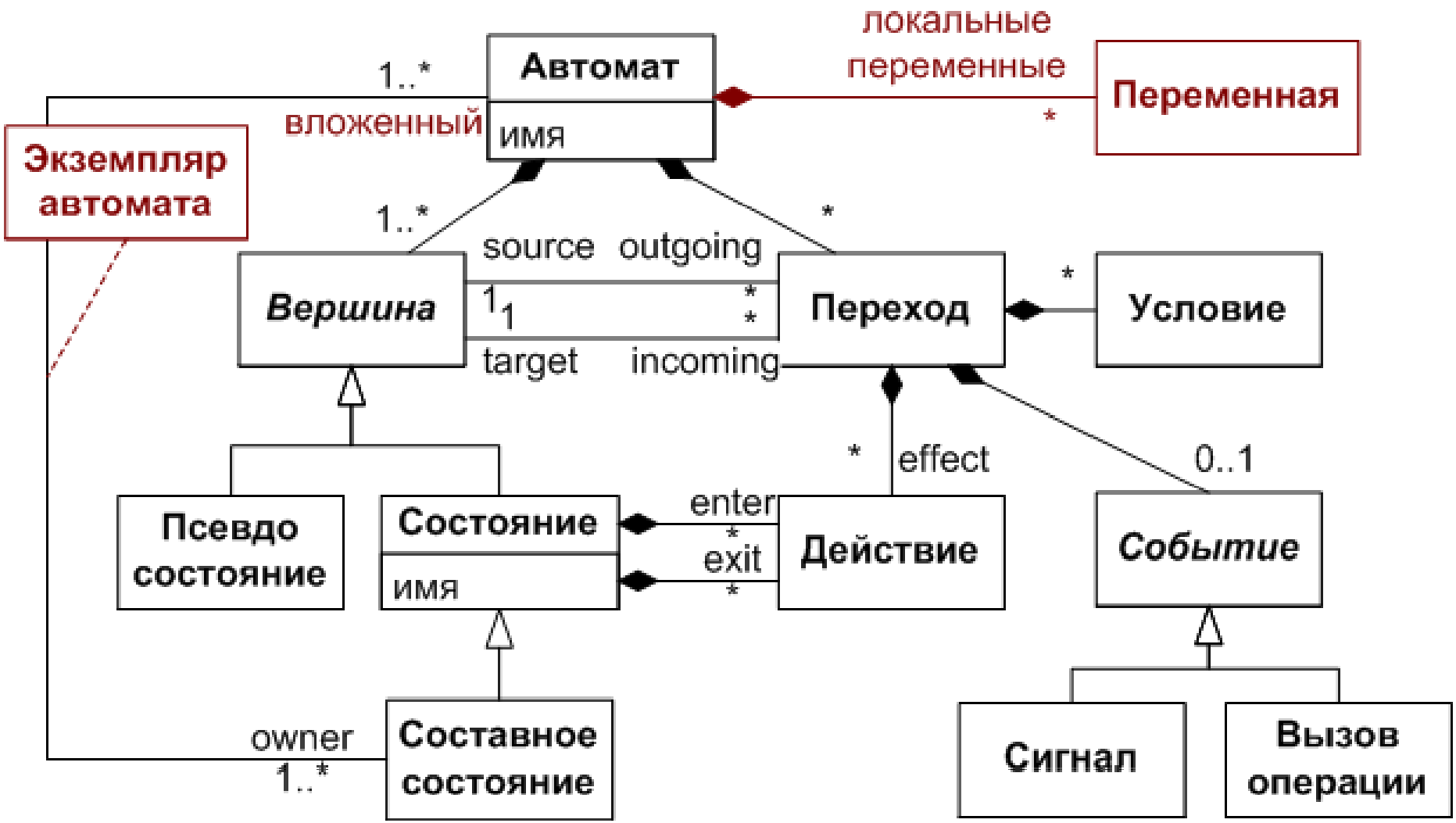
Схема применения определения языка



Используемая автоматная модель



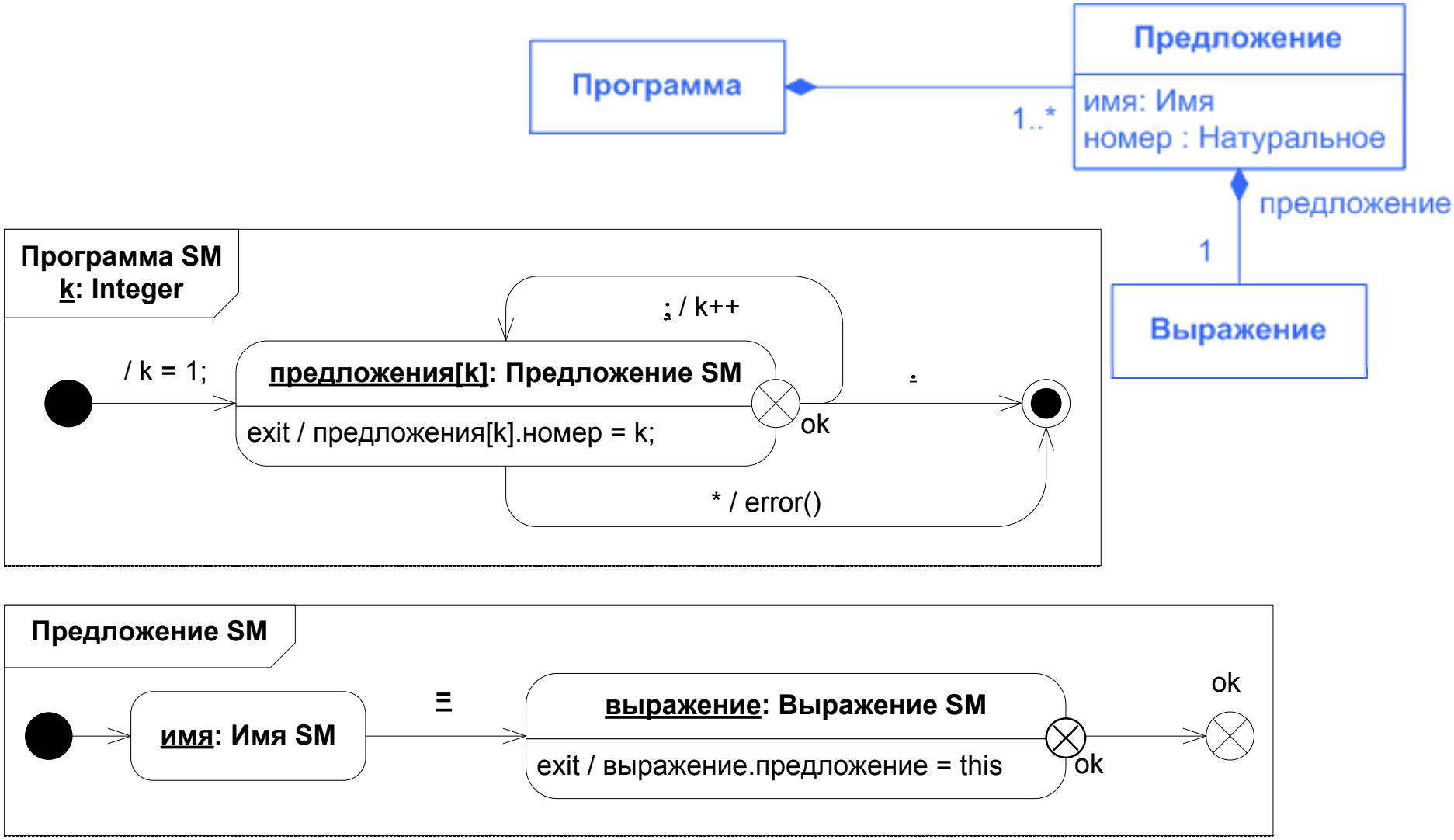
Метамодель автомата



Методика построения распознающих автоматов по метамодели языка

- α Начальной структурной единице (аксиоме) языка соответствует головной автомат системы
- β Каждому классу с именем A в метамодели языка соответствует класс автоматов с именем A SM. Экземпляр автомата A SM в процессе выполнения строит экземпляр класса A, то есть является его конструктором
- γ Каждой составляющей класса A, которая имеет неэлементарный тип B в автомате A SM соответствует составное состояние, в которое вложен автомат B SM
- δ Альтернативной декомпозиции классов соответствует альтернативная декомпозиция автоматов, а именно: в каждое составное состояние вкладывается отдельный экземпляр автомата указанного класса
- ε Дизъюнктивной композиции классов, перечислимым типам и обобщению классов соответствуют сегментированные переходы на диаграмме автомата
- ζ Кратности полюса (или атрибуту-массиву) соответствует петля на диаграмме автомата и массив вложенных экземпляров автоматов

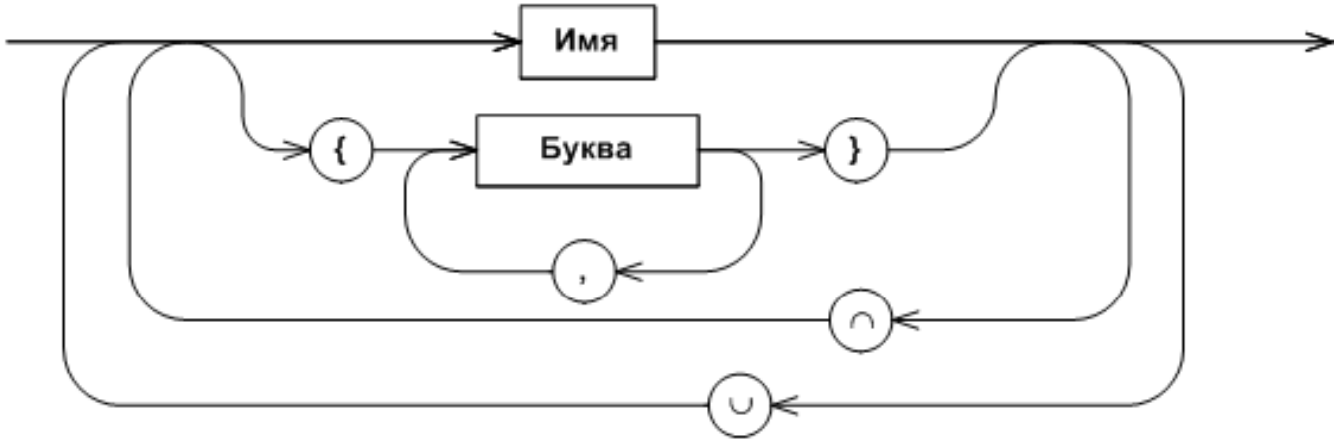
Головной и основной автоматы для мини языка множеств



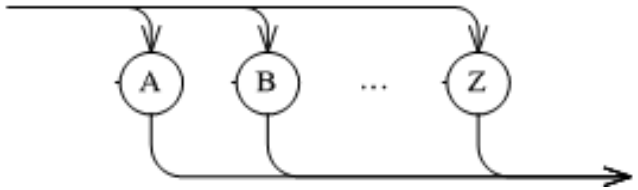
Использование синтаксических диаграмм

Программа

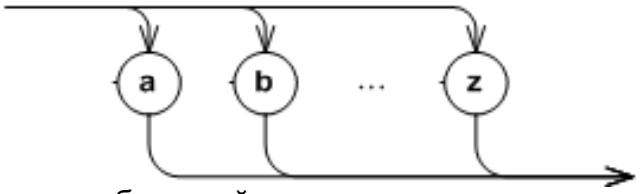
Выражение



Имя



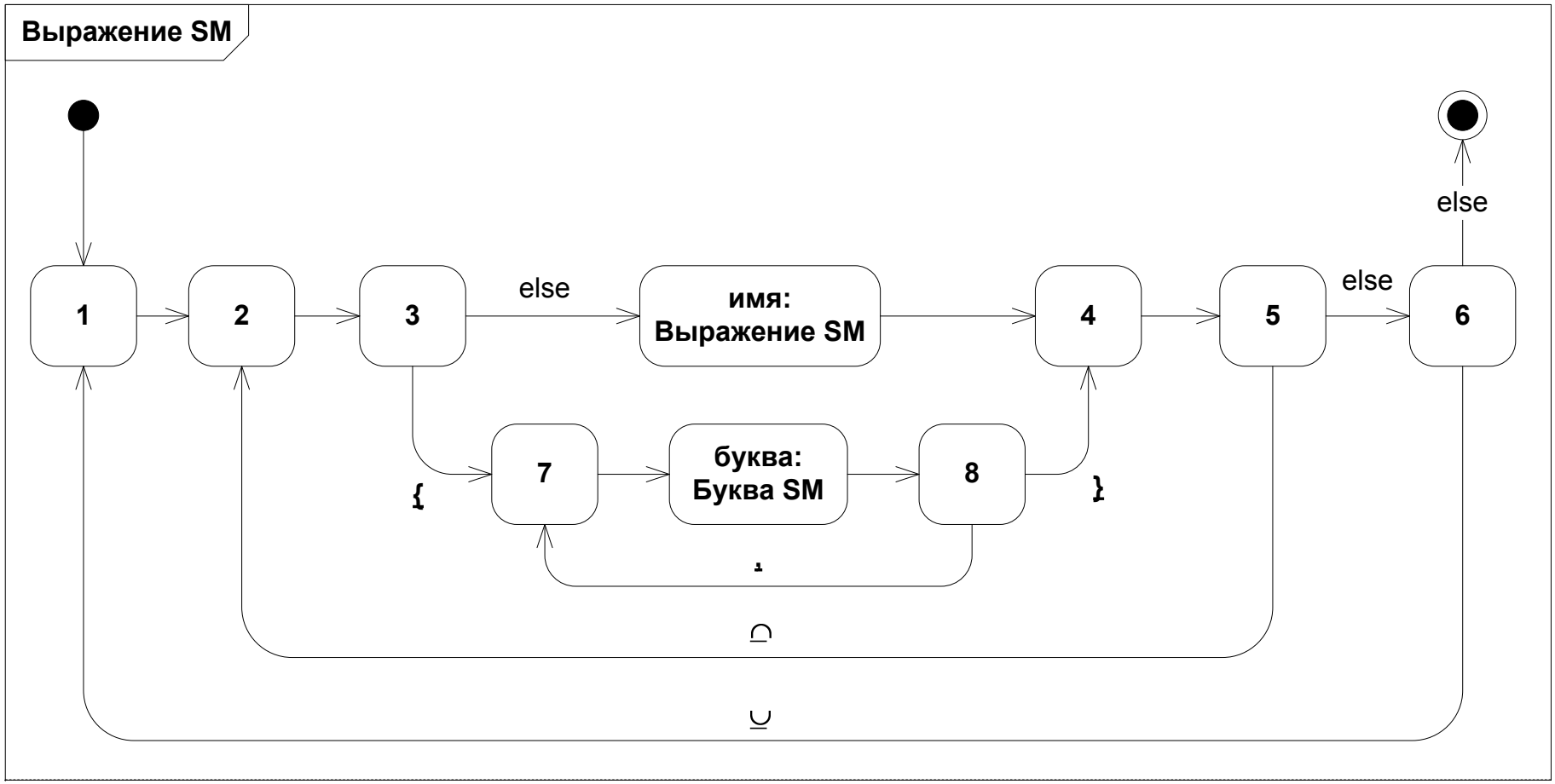
Буква



Методика преобразования синтаксических диаграмм в систему автоматов

- α Каждая синтаксическая диаграмма преобразуется в диаграмму автомата
- β Вводится начальное состояние, к которому присоединяется входная стрелка, и заключительное состояние, к которому присоединяется выходная стрелка
- γ Нетерминалы преобразуются в составные состояния, в которые вложены автоматы соответствующих конструкций
- δ В каждой точке разветвления или слияния дуг синтаксической диаграммы вводится служебное состояние
- ε Терминалы переносятся на дуги перехода в качестве событий
- ζ Немотивированные переходы, альтернативные мотивированным, помечаются ключевым словом else

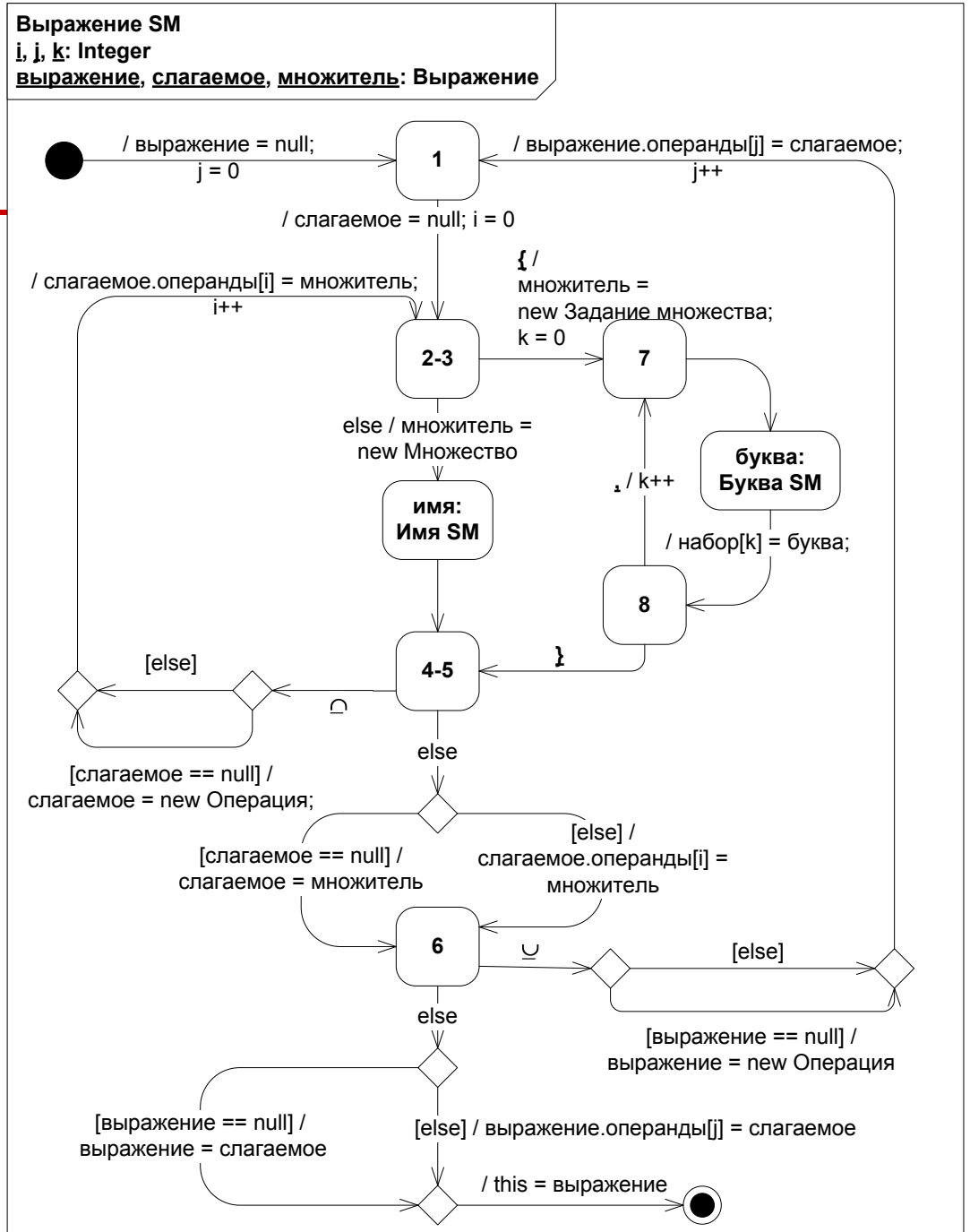
Автоматически построенный автомат распознаватель



Вручную доработанный автомат конструктор

абстрактный синтаксис целесообразно рассматривать как главное и первичное

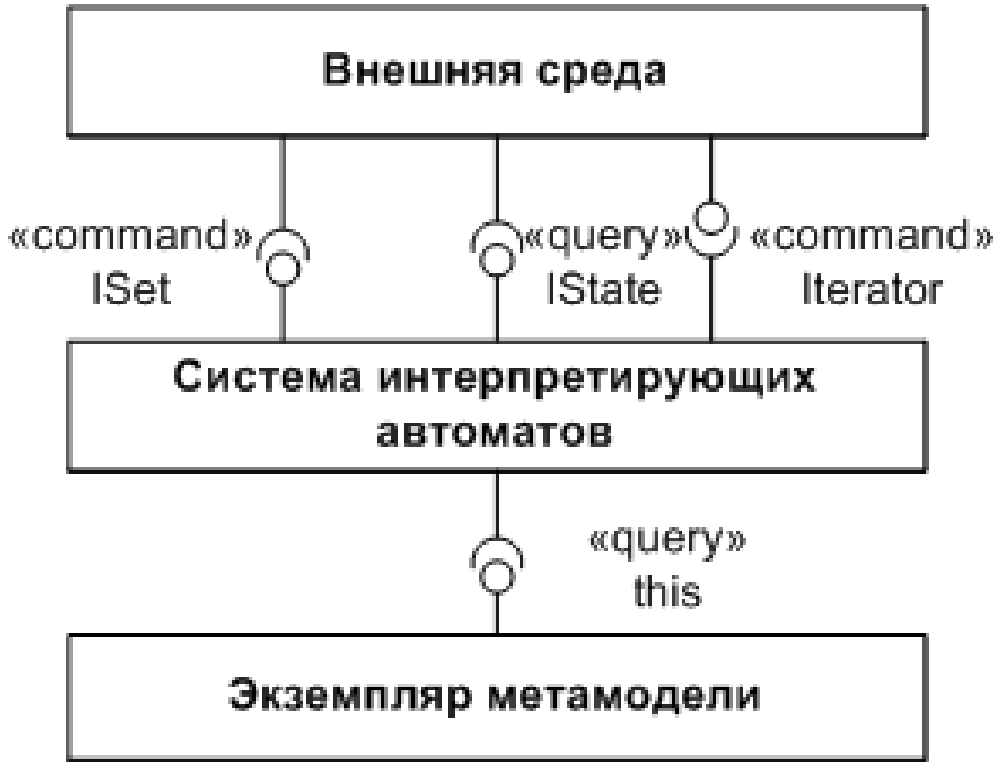
описание языка

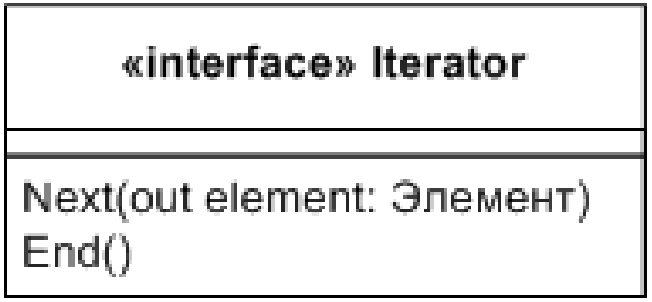


Задание семантики в автоматном методе

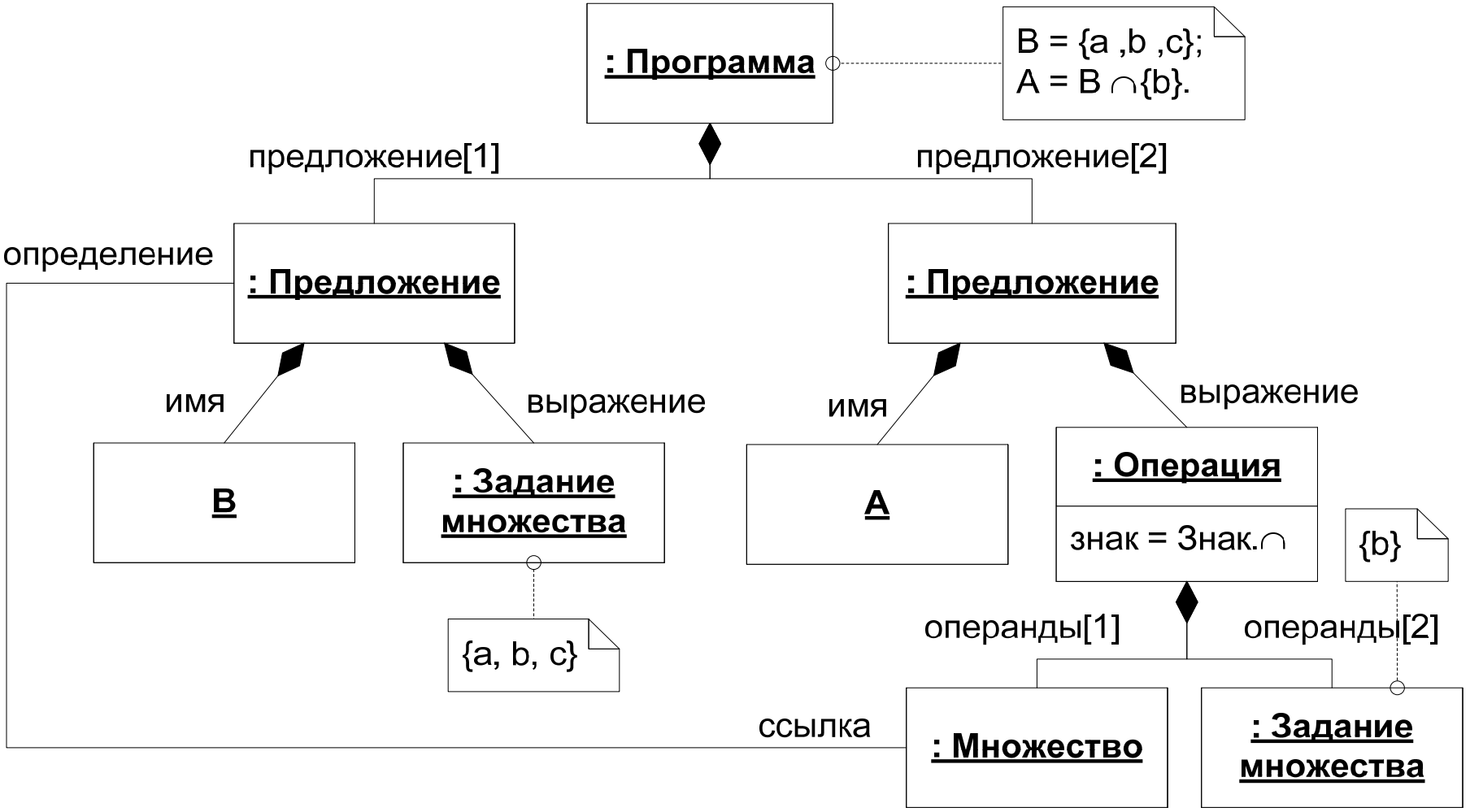
Тип семантики	Область применения	Структуры данных (для мини языка множеств)
Преобразование входа программы в её выход	Системы пакетной обработки	Выход программы = M [Имя, Буква]: Boolean
Последовательность побочных эффектов в процессе выполнения программы	Интерактивные системы управления	
Сервис или служба, отвечающая на запросы пользователя	Интеллектуальные экспертные системы	B = {a, b, c} A = B ∩ {b} A = ?

Схема взаимодействия автоматов семантики





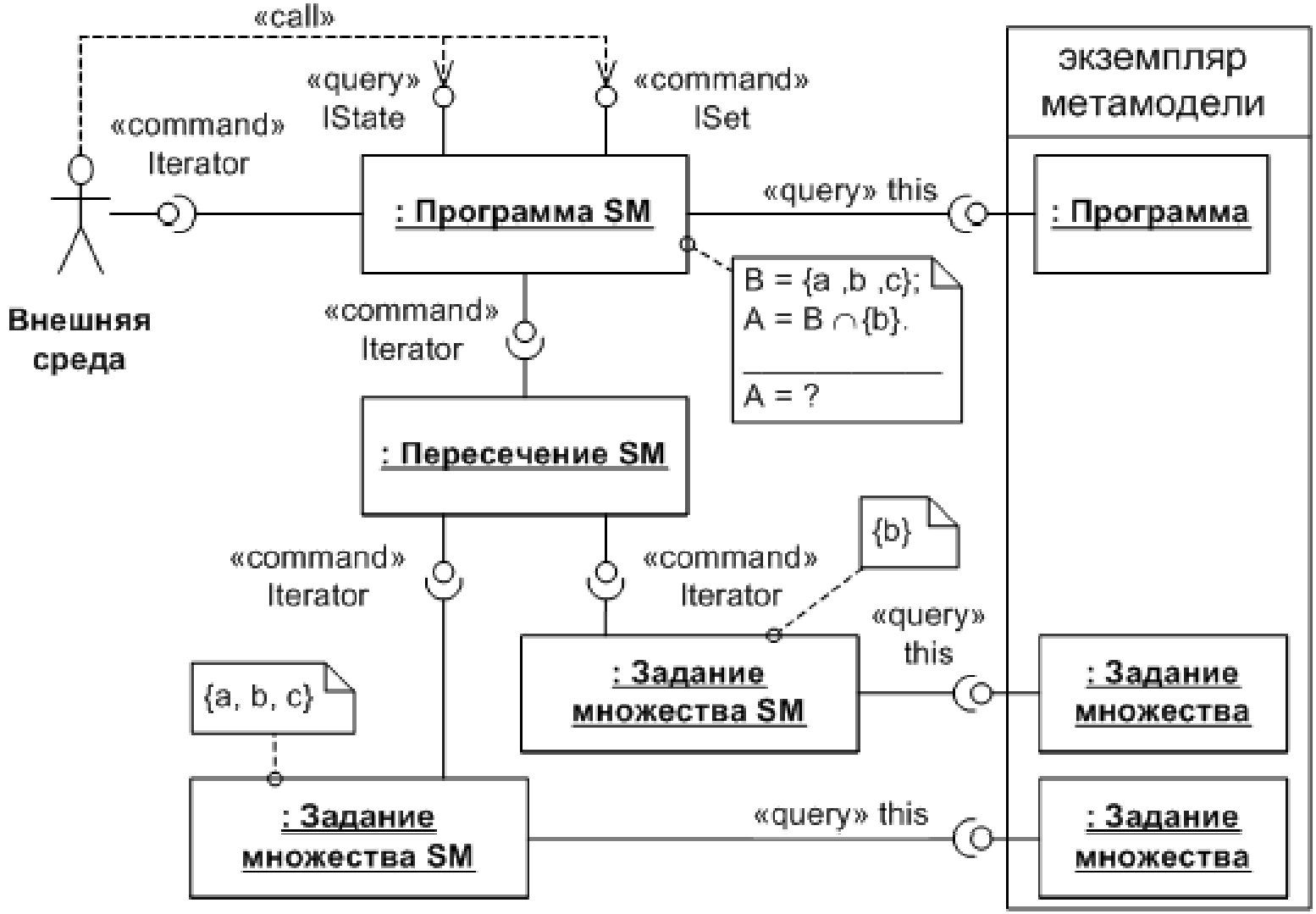
Экземпляр метамодели (абстрактная программа)



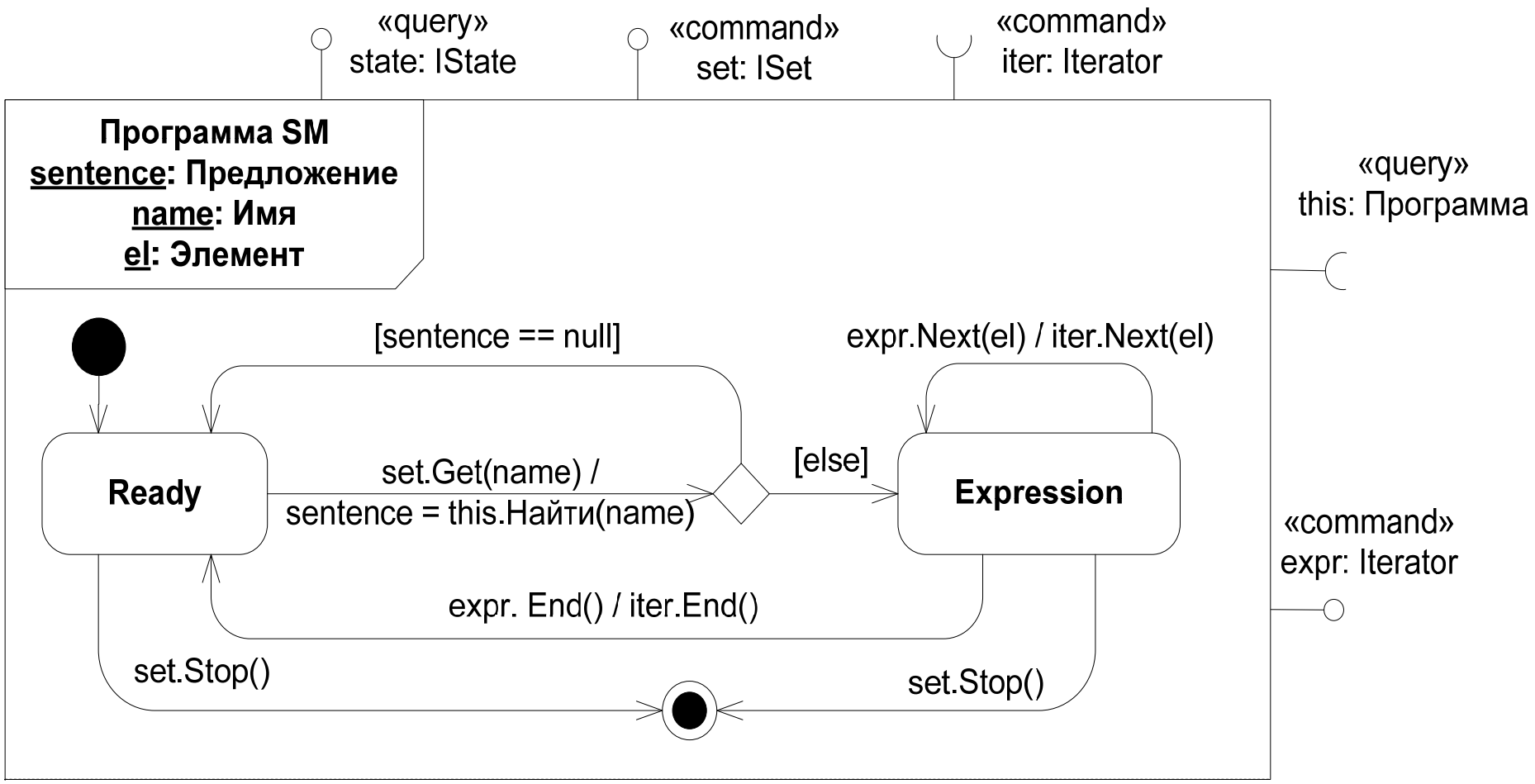
Правила трансформации абстрактной программы в систему взаимодействующих автоматов

```
proc CreateAuto (expr: Выражение, parent: Iterator) : Выражение SM
    var result: Выражение SM
    case expr.GetType() of
    typeof (Задание множества):
        result = new Задание множества SM()
        result.this = expr; result.iter = parent
    typeof (Множество):
        result = CreateAuto (expr.определение.выражение, parent)
    typeof (Операция):
        if expr.знак == ∩ then      result = new Пересечение SM()
        else // expr.знак == ∪ //   result = new Объединение SM()
        endif
        result.left  = CreateAuto (expr.операнды[1], result)
        result.right = CreateAuto (expr.операнды[2], result)
        result.iter  = parent
    end case
end proc
```

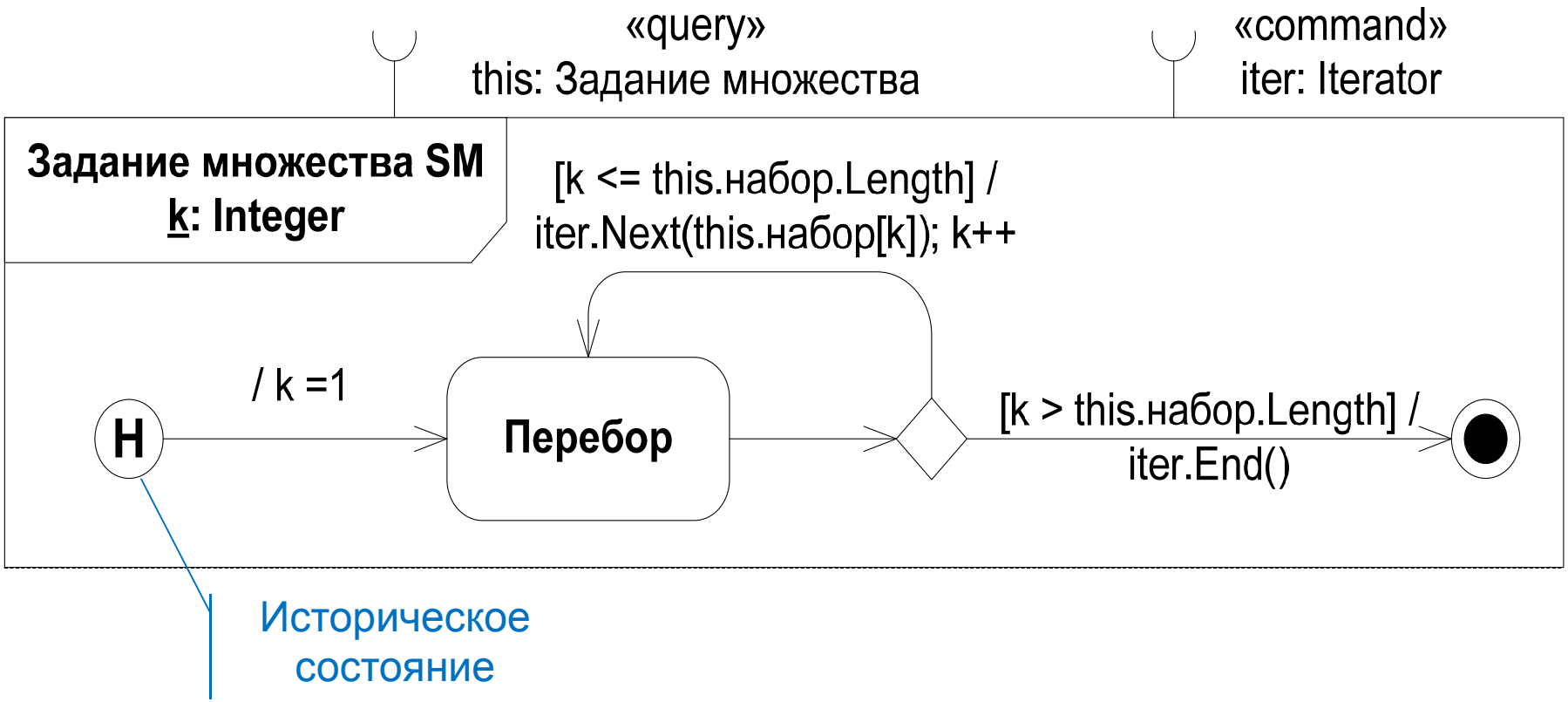
Взаимосвязь экземпляров автоматов семантики



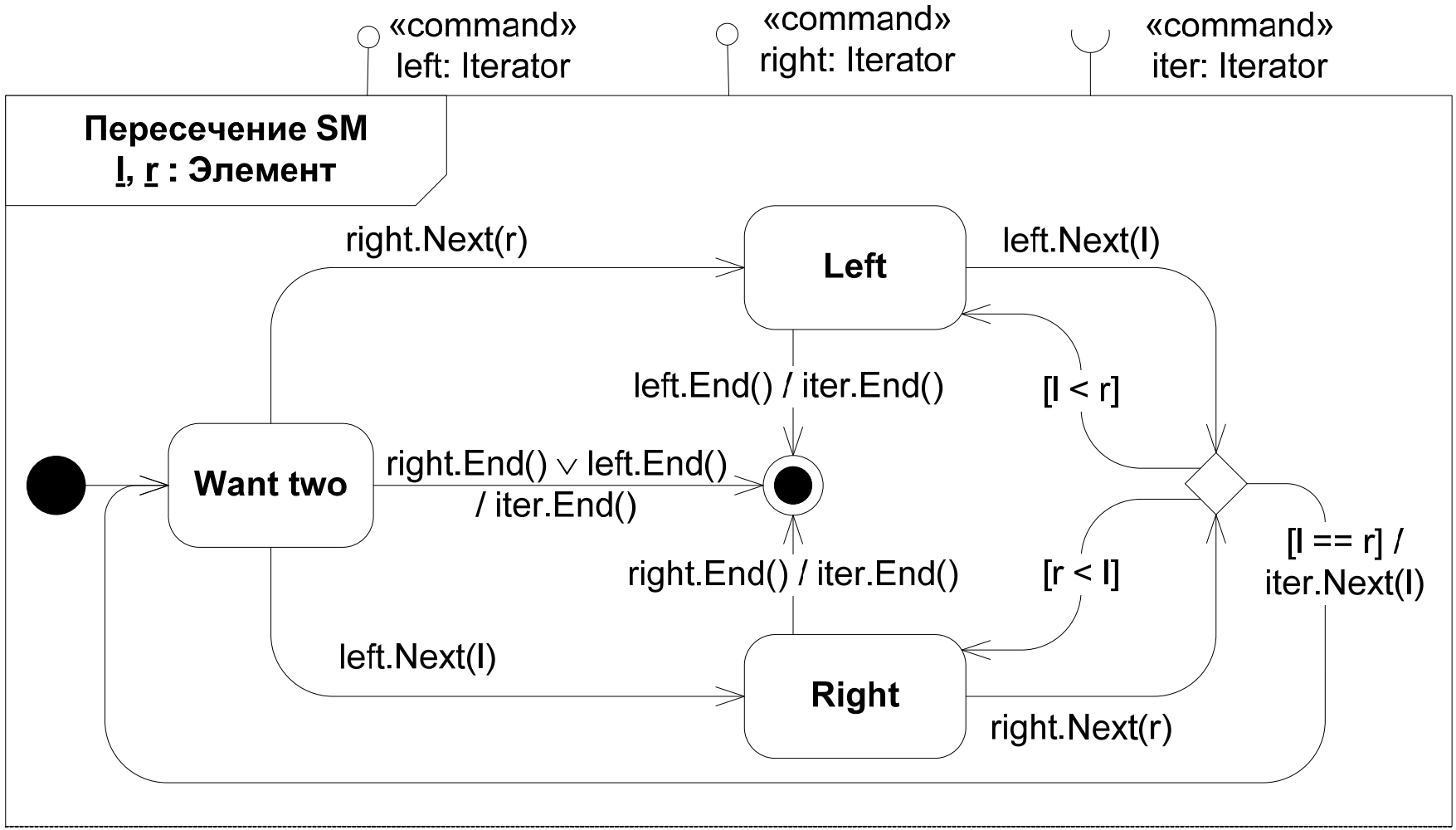
Головной автомат семантики мини языка множеств



Автомат интерпретации задания множества



Автомат интерпретации задания множества



Модель использования автоматного метода

Конкурентные преимущества и открытые вопросы

+ Расширение абстрактного синтаксиса и включение в метамодель неиерархических отношений
+ Унифицированное описание конкретного синтаксиса и семантики
+ Включение в автоматную модель всех четырех возможных способов взаимодействия автоматов
+ Естественный параллелизм автоматных программ
+ Соответствие международному стандарту UML

? Описание класса контекстных условий, допускаемых автоматным методом

? Описание трансформаций, получающих систему автоматов из метамодели

? Преобразование автоматных программ в другие модели описания поведения

? Практическое использование автоматной параллельности

? Определение необходимых расширений и профилей UML

Выводы по главе 3

- Автоматный метод определения языков программирования позволяет
  - дать полное формальное определение любого языка, в том числе и не текстового
  - по формальному определению автоматически получить готовую реализацию
- Формальное определение включает в себя
  - определение метамодели языка, в том числе контекстных условий, которые невозможно задать порождающими грамматиками
  - конкретный синтаксис и операционную семантику языка, заданные системами взаимодействующих автоматов
- Предложена автоматная модель, обеспечивающая гибкое конструирование систем взаимодействующих автоматов
- Предложены системы правил и методики полуавтоматического получения автоматов конкретного синтаксиса и автоматов семантики на основе метамодели языка
- **Автоматный метод определения проблемно-ориентированных языков** является третьим положением, выносимым на защиту

Глава 4. Методы автоматического синтеза программ, основанные на декларативных языках спецификации

- **Задача главы**: представление и развитие методов определения моделей предметных областей, средств постановки задач и предметно-ориентированных языков, допускающих автоматический синтез программ
- **План главы**:
  - ***Параграф 4.1. Язык исполняемых программных спецификаций***
    - Эпиграф: Язык Ψ = PSI = Program Specification Interpreter = Program Systems Institute
  - ***Параграф 4.2. Исчисление структурного синтеза***
    - Эпиграф: расширение исчисления сужает класс его моделей
  - ***Параграф 4.3. Алгоритмы структурного синтеза программ***
    - Эпиграф: чем сложнее предобработка модели, тем проще ее использование

История вопроса

1970 Тыугу «Решение задач на вычислительных моделях», ЖВМ и МФ

1977 Тыугу «Система программирования ПРИЗ», ИК, Таллин

1980 Тыугу, Харф «Алгоритм структурного синтеза программ», Программирование

1980 Бабаев, Новиков, Петрушина «Язык Декарт – входной язык системы СПОРА», Прикладная информатика

1982 Минц, Тыугу «Полнота правил структурного синтеза», Докл. АН СССР

1983 Новиков «Реализация абстрактных типов и отображений в программном обеспечении прикладных исследований», диссертация

1985 Диковский «Решение в линейное время алгоритмических проблем связанных с синтезом ациклических программ», Программирование

1987 Лавров «*D*-аксиоматизация языка Декарт», Прикладная информатика

2006 Новосельцев «Теория структурных функциональных моделей», Сибирский математический журнал

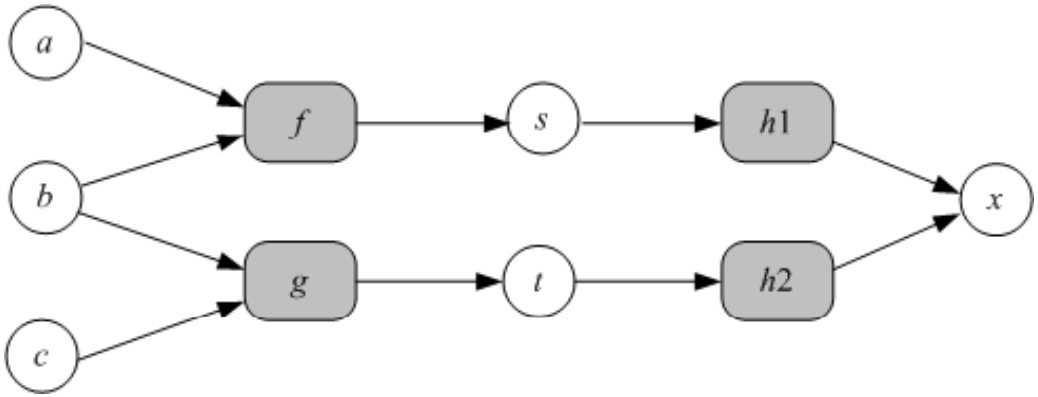
2010 Новиков, Новосельцев «Язык исполняемых программных спецификаций», Программирование

Принципы языка Ψ

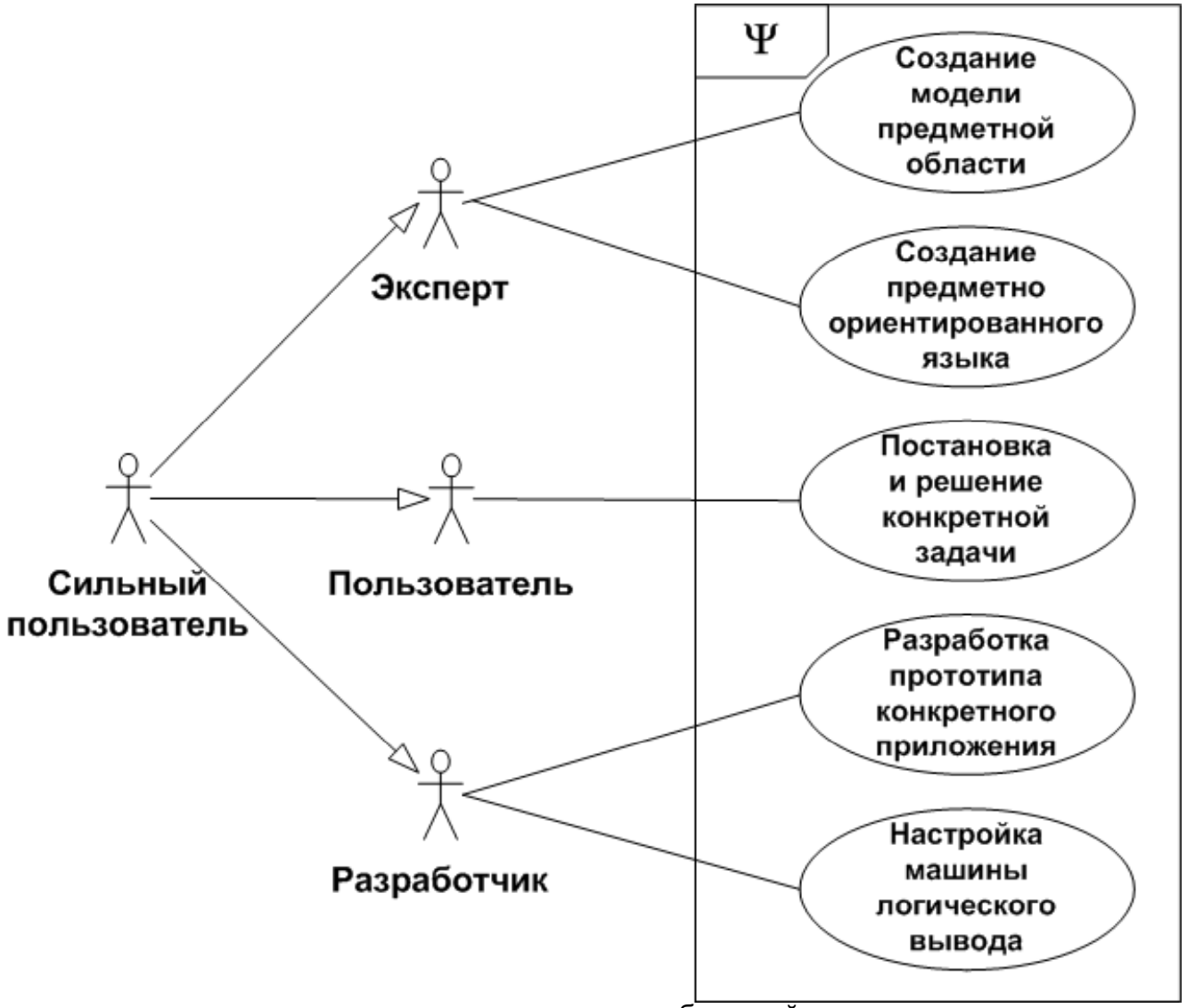
- Ориентация на декларативную спецификацию предметной области в терминах формальной теории в специальном логическом исчислении
- Симметричное описание структур (сущностей) и поведения (связей) конкретной задачи в данной предметной области: сущности определяются через связи, а связи определяются через сущности
- Автоматический синтез схемы решения задачи с последующей компиляцией или интерпретацией схемы на основе готового программного фонда предметной области
- Открытое определение метамодели языка и предоставление пользователю возможности расширения и изменения метамодели
- Определение набора конкретных синтаксисов, однозначно отображаемых в мета-модель, в том числе графической нотации

Область применения языка Ψ

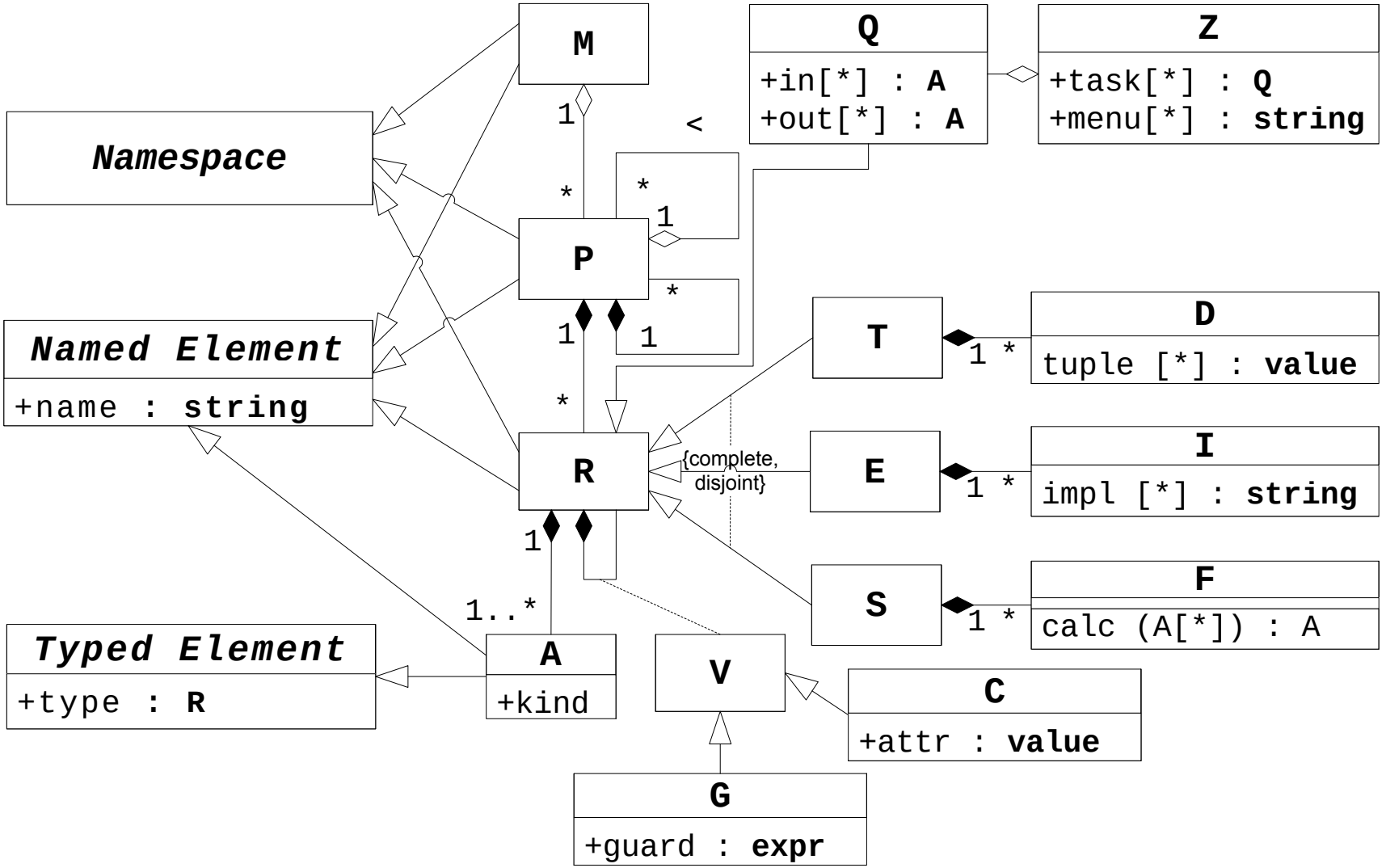
- Семантическая вычислительная сеть (МПО) $M = <V, F>$
  - Переменные V + Связи $F = \{ f \mid f: V \times \ldots \times V \rightarrow V \}$
  - Двудольный граф (без ограничений)
- Непротиворечивость постулируется
- Реализации связей считаются заданными
- Задача: $T = <A, X>, A \subset V, X \subset V \cong X \subset F^*(A)$
- Модели с условиями: предикаты на вершинах обеих долей



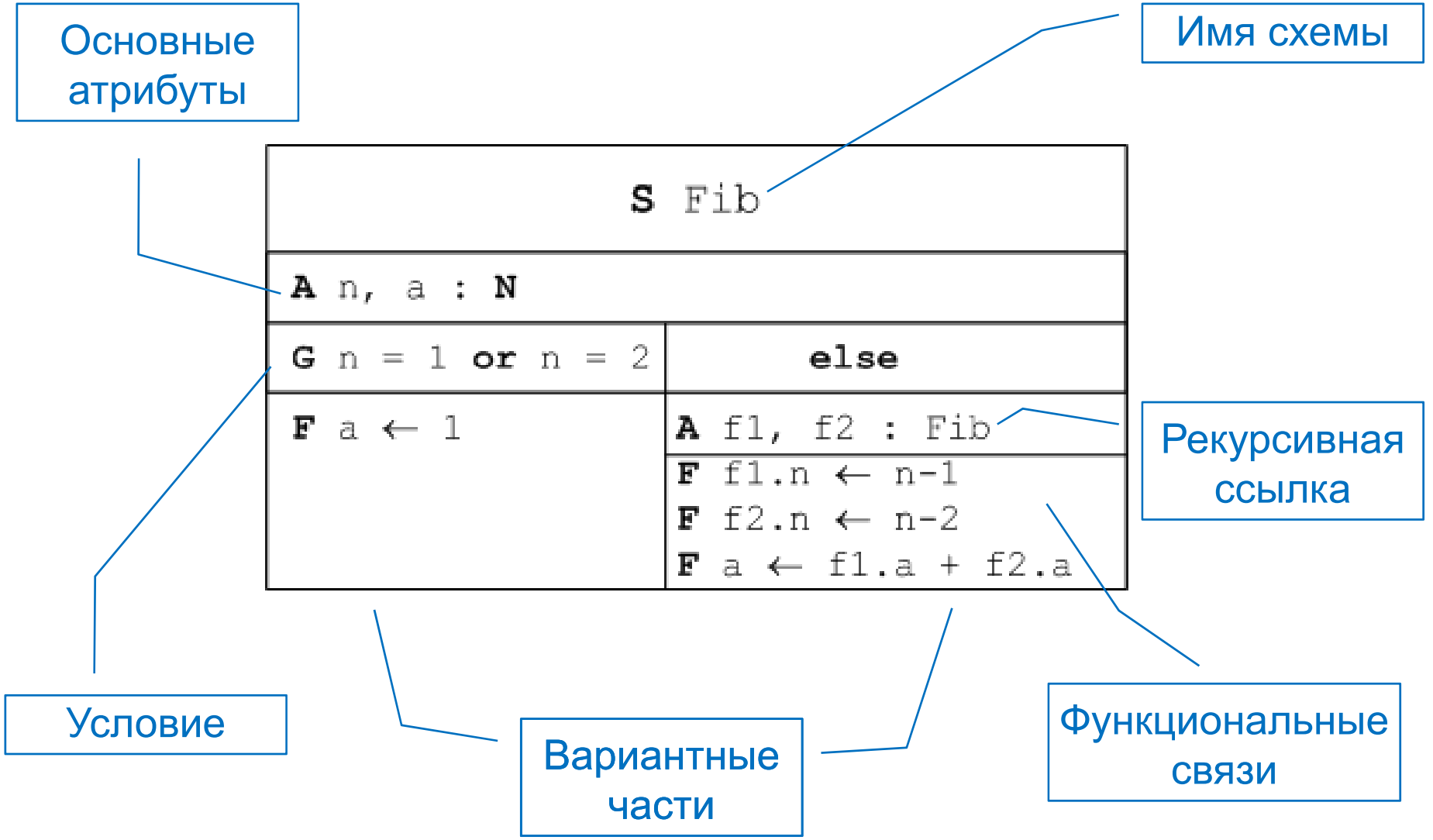
Модель использования языка Ψ



Метамодель языка Ψ



Пример описания схемы: числа Фибоначчи



Пример описания схемы: треугольник

1. **Package** Geometry { **Scheme** Triangle {
2. **Real** a, b, c, α, β, γ, s, p |
3. p ← 0.5 * (a + b + c);
4. s ← sqrt (p * (p – a) * (p – b) * (p – c)) }
5. **if** α = π/2 **then**
6. a ← sqrt (b*b + c*c);
7. **else**
8. a ← sqrt (b*b + c*c – 2*b*c*cos(α));
9. **fi**
10. **Q** find_s (**on** Simple **in** a, b, c **out** s)
11. **Q** find_a (**on** Simple **in** α, b, c **out** a) }

Пакет и схема

Атрибуты

Связи

Вариантные части

Постановки задач

Язык Пси

```
P Simple {
  S Simple {
    int a, b, c, d, e     |
    a <- b + c;
    c <- e * 2  }
  if c > 0 then {
    int i      |
    i <- c * c;
    b <- c * 2 * i   }
  else
    b < – c * 3
  fi;
  Q find_a {on Simple in e out a}
}
```

Язык Си

```
#include <stdlib.h>
int find_a(int e) {
  int i;  int b;  int c;  int a;
  if ((c > 0)) {
    i = (c * c);
    b = ((c * 2) * i);  }
  else {
    b = (c * 3);   }
  c = (e * 2);
  a = (b + c);
  return a; }
int main (int argc, char *argv[]) {
  int a;
  a = find_a (int e);
  return 0;
}
```

Формальная теория языка Ψ

- A – множество имен атрибутов, F – множество имен функций
- $\mathrm{Arg}(f)\subset A$, $\mathrm{Res}(f)\subset A$, $f\in F$ – аргументы и результаты функции
- $<A, F>$ – плоская модель предметной области
- $C(X, Y, G)$, $X\subset A$, $Y\subset A$, $G = (g_1, \ldots, g_n)$, $g_i\in F$ – предложение вычислимости
- $\forall\ Y\subset X\ C(X, Y, (\ )\ )$ – предметно независимая схема аксиом
- $\forall\ f\in F\ C(\mathrm{Arg}(f), \mathrm{Res}(f), (f)\ )$ – множество предметных аксиом
- $C(X, Y, (G_1)\ )\ \&\ C(W, Z, (G_2)\ )\ \&\ W\subset Y \Rightarrow C(X, Y\cup Z, (G_1; G_2)\ )$ – правило композиции
- $Q = <A, F, X, Y>$, $X\subset A$, $Y\subset A$ – задача на плоской модели
- $\exists\ G\ \exists U\subset X\ \ \exists Y\subset W\ C(U, W, G)$ – решение G задачи $Q = <A, F, X, Y>$
- **Теорема** (О полноте правила композиции). *Пусть* $G=\{g_1, \ldots, g_n\}$ – *решение задачи* $Q = <A, F, X, Y>$. *Тогда оно может быть получено применением правила композиции*.

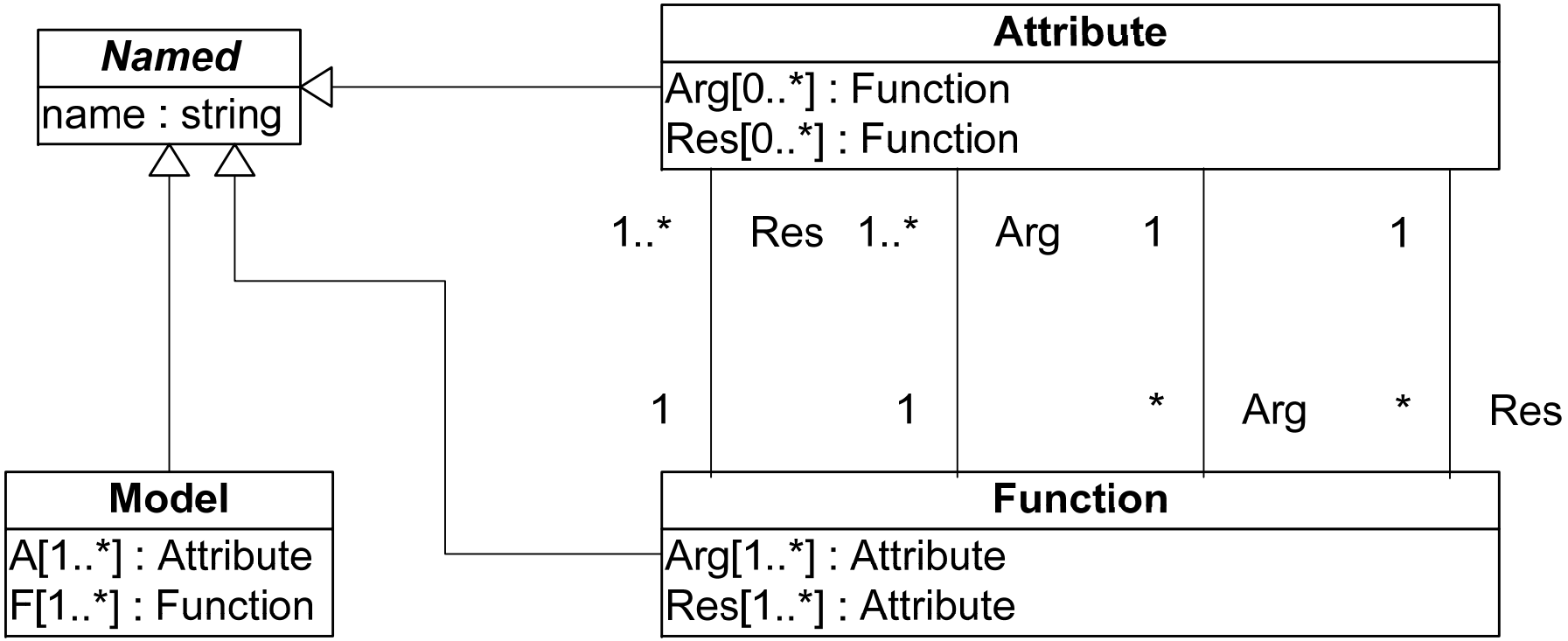
Волновой алгоритм синтеза линейных программ на плоской модели (1 из 2)

1. U := X // список известных атрибутов
2. W := Y \ X // список неизвестных атрибутов
3. G := [] // пустая программа
4. **while** W≠∅ **do**
5. g := **nil** //функция применима
6. **for** f ∈ F **do**
7. **if** (Arg(f)⊂U) & (Res(f)⊄U) **then**
8. g = f
9. **exit for**
10. **end if**
11. **end for**
12. **if** g = **nil then**
13. **return fail**
// задача неразрешима
14. **else**
15. U := U ∪ Res(g)
16. W := W \ Res(g)
17. F := F – g
18. G := G + g
19. **end if**
20. **end while**

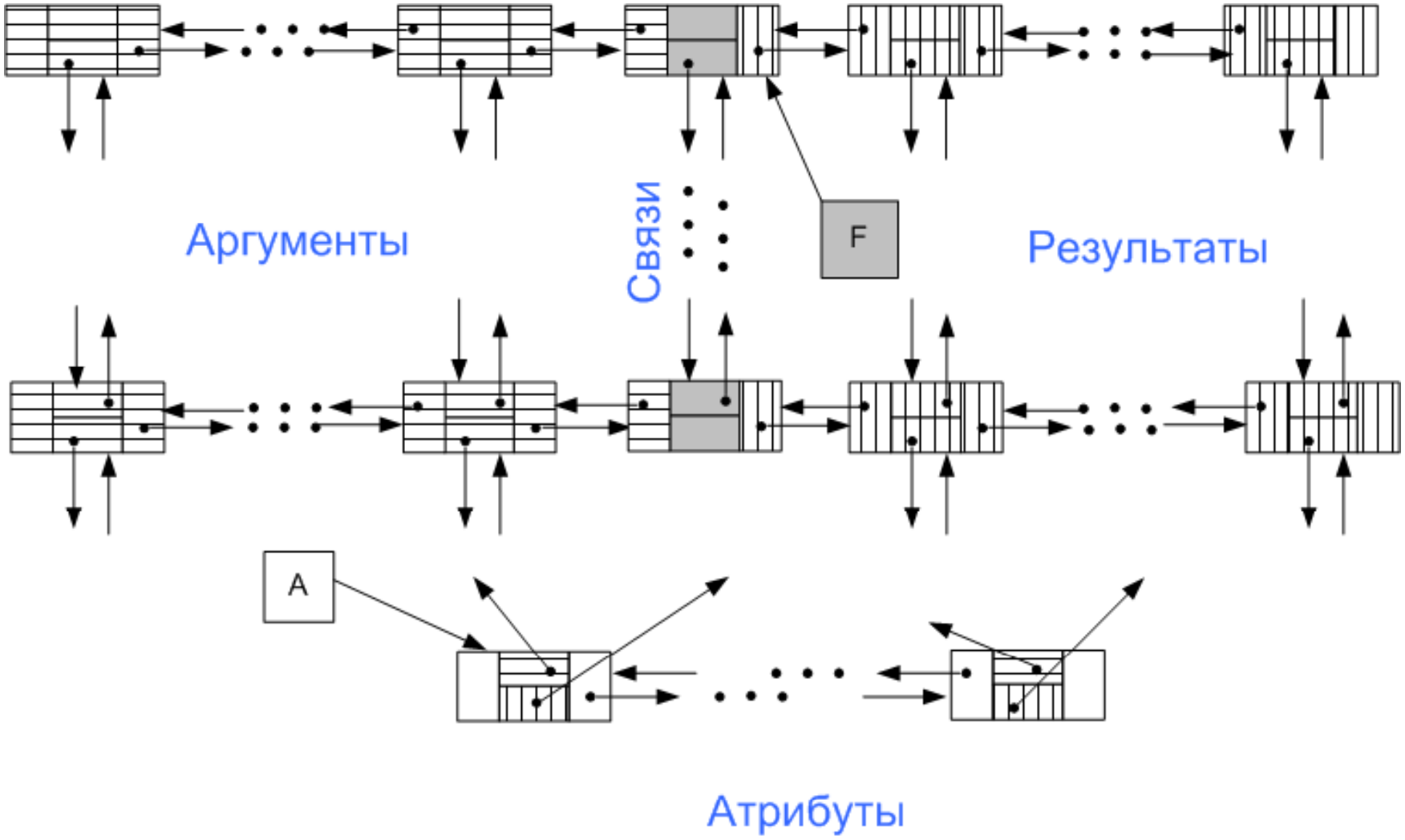
Волновой алгоритм синтеза линейных программ на плоской модели (2 из 2)

1. H := [] // «очищенная» программа
2. T := Y // множество «необходимых» атрибутов
3. G := Reverse(G) // обращение списка
4. **for** g ∈ G **do** // просмотр списка в обратном порядке
5. **if** Res(g) ∩ T ≠ ∅ **then** // функция нужна
6. T := T \ Res(g) //
7. T := T ∪ Arg(g) //
8. H := H + g //
9. **end if**
10. **end for**
11. H := Reverse(H) // восстанавливаем порядок

Предобработка модели предметной области (1 из 2)



Предобработка модели предметной области (2 из 2)



Линейный алгоритм синтеза программ

```
1.  proc  Solve (A, F, X, Y)
2.  H := ∅    // множество применимых связей
3.  G := []    // синтезируемая программа
4.  U := Y \ X  // список целевых атрибутов
5.  for a ∈ X do Process(a) end for // дано
6.  while U ≠ ∅ do
7.     if H = ∅ then return fail end if // неудача
8.     select   g ∈ H
9.     G := G + g
10.    U := U \ g.right
11.    for a ∈ g.right do
12.       Process(a)
13.    end for
14.  end while
15. end proc
16. // процедура удаления ячейки
17. proc Del (v)
18.    v.left .right := v.right; v.right.left := v.left;
19.    v.up.down:= v.down; v.down.up := v.up;
20.    Dispose(v)
21. end proc
```

```
1.  proc Process(a)
2.    for s ∈ a.up do // аргументы
3.      if s.left = ∅ & s.right ∈ F then
4.          g := s.right // последний аргумент
5.          H := H + g // связь применима
6.          for r ∈ g.right do Del(r) end for
7.          Del (g)
8.       end if
9.       Del (s)
10.    end for
11.  for s ∈ a.down do // результаты
12.    if s.right = ∅ & s.left ∈ F then
13.        g := s.left // последний результат
14.        H := H – g // связь бесполезна
15.        for r ∈ g.left do Del(r) end for
16.        Del (g)
17.      end if
18.      Del (s)
19.    end for
20.    Del (a)
21. end proc
```

Обзор деталей, опущенных в докладе

1. Условные предложения вычислимости $CC(X, Y, G, Q)$ и синтез программ с ветвлениями
2. Синтез параллельных программ для платформы T++
3. Использование иерархии отношений без развертки и синтез программ со структурами (**struct**)
4. Выделение в языке Пси подъязыков описания модели, описания ситуации и постановки задач
5. Синтез рекурсивных программ
6. Синтез циклических программ по системам рекуррентных соотношений

Выводы по главе 4

- Автоматический синтез вычислительных программ по декларативным спецификациям — перспективный метод алгоритмизации и формализации предметных областей
- Структурный синтез программ имеет четко очерченную область эффективного применения — предметные области, допускающие адекватные модели в форме семантических вычислительных сетей
- Применение формальных (логических) методов совместно с программными эвристиками позволяет получать полезные результаты
- **Язык исполняемых программных спецификаций** и методы его реализации — четвертое положение, выносимое на защиту

Заключение: внедрение полученных результатов и положения, выносимые на защиту (1 из 7)

- Использование разработанных автором DSL для повышения качества алгоритмизации вычислительных предметных областей в учреждениях Академии наук
- Внедрение разработанных автором методов повышения качества алгоритмизации в промышленности при автоматизации бизнес-процессов и создании DSL
- Применение разработанных автором концепций повышения качества алгоритмизации в университетах в учебном процессе и подготовке кадров
- Методы повышения качества алгоритмизации
- Положения, выносимые на защиту
- Избранные публикации по теме диссертации

Заключение: (2 из 7) использование в академических институтах

- Система СПОРА 1978-1986
  - Протокол приемо-сдаточных испытаний Межведомственной комиссии при Совмине СССР
  - Свидетельство ГосФАП СССР, №50860000399
- Система ЭРА 1986-1992, 2007-2010
  - Научные отчеты ИТА АН СССР, ИПА АН СССР, ИПА РАН
- Система АстроТОП 1991-1997
  - Научные отчеты ИТА РАН, ИПА РАН
- Система Дельта 2005, 2010
  - Научные отчеты ИПА РАН
- Система СВИТА 1993-1999
  - Научные отчеты ИТА РАН, ИПА РАН
- Проект СКИФ-ГРИД 2009-2010
  - Научные отчеты ИПС РАН по проекту 2010р424 «Инструментальный программный комплекс моделирования сложных предметных областей с возможностями автоматического синтеза параллельных программ»

Заключение: (3 из 7) внедрение в промышленности

- ASDH 1997–2000, АстроСофт
  - Регламенты проведения типового проекта по разработке программного обеспечения, внутренняя документации компании АстроСофт
- UniMod, ИТМО, 2005-2007
  - Результаты конкурса ФЦНТП «Исследования и разработки по приоритетным направлениям развития науки и техники на 2002-2006 годы». Извещение №9-к-663 от 2 мая 2005 г.
  - Отчеты по госконтракту ИТ-13.4/004 «Технология автоматного программирования: применение и инструментальные средства»
- аСМК, АтДиа, 2007-2008
  - Результаты конкурса фонда Бортника «СТАРТ 07»
  - Отчеты по проекту «Разработка инструментальных программных средств на базе © MS Office для внедрения автоматизированной Системы Менеджмента Качества (СМК) в компаниях с проектным типом организации производства»
  - Свидетельство об официальной регистрации программы для ЭВМ №2007612840 от 27 июня 2007 года «Конструктор системы менеджмента качества на базе MS Office»

Заключение: (4 из 7) применение в учебном процессе

- Курс «Дискретная математика для программистов»
  - Учебник «Дискретная математика», 2011 ← «Дискретная математика для программистов», три издания, 2000, 2003, 2008
    - СПБГУ с 1986 года, ИТМО с 2010 года,
- Курс «Моделирование на UML»
  - Монография «Моделирование на UML» (с Ивановым Д.Ю.), 2010
  - Sun Microsystems Teaching Grant 2008, «Анализ и проектирование на UML»
    - СПбГПУ с 2000 года, ИТМО с 2005 года, мат/мех 2006-2007, семинары в Интернете, корпоративные курсы в организациях
- Курс «Системы представления знаний»
  - Монография «Искусственный интеллект: представление знаний и методы поиска решения» (с Новосельцевым В.Б.), 2011
    - СПбГПУ с 1992 года

Заключение: (5 из 7) подготовка квалифицированных кадров (всего 64)

Год	Степень	Фамилия	Тема
1995 ИТА	К.ф.-м.н.	Крашенинников С.В.	Управление данными в системе таблично-ориентированного программирования
2003 СПбГПУ	Магистр	Степанян К.Б.	Автоматическое представление модели UML
2007 ИТМО	Магистр	Лукьянова А.П.	Преобразования программ, сохраняющие поведение
2008 СПбГПУ	Магистр	Тихонова У.Н.	Определение языков программирования интерпретируемыми автоматами
2009 ИТМО	Диплом	Клебан В.О.	Автоматизация документооборота с использованием конечных автоматов
2010 ИПА	К.ф.-м.н.	Михеева В.Д.	Решение задач эфемеридной астрономии средствами предметно-ориентированного языка программирования
2010 СПбГПУ	Бакалавр	Фишков А.А.	Анализ и реализация методов синтеза программ на модели предметной области

Заключение: (6 из 7) методы повышения качества алгоритмизации

- Определение DSL, обладающего достаточной выразительной силой для описания решения всех типовых задач предметной области
  - Определение языка должно быть достаточно формализованным, в основе определения должна лежать метамодель
  - Класс типовых задач должен быть достаточно представительным
- Построение (мета) модели предметной области, достаточно полной и детальной для представления знаний о решении типовых задач
  - Целесообразно использовать формализмы искусственного интеллекта
  - Уровень абстракции понятий модели предметной области следует выбирать «с запасом»
- Вовлечение сильного пользователя, способного решить любую типовую задачу и знающего ответы на все вопросы о предметной области
  - Качество растет, если сильный пользователь является автором (соавтором) DSL и МПО
  - Сильный пользователь может быть: человеком, комитетом, сводом законов, великой книгой, …

Заключение: (7 из 7) положения, выносимые на защиту

- Предложена **концепция циклов повышения продуктивности** для инкрементных и спиральных моделей процесса разработки прикладного программного обеспечения, объясняющая механизм влияния проблемно-ориентированных языков на продуктивность разработки
- Предложена **парадигма таблично-ориентированного программирования**, реализовано и внедрено семейство таблично-ориентированных языков в наукоемкой предметной области «эфемеридная астрономия»
- Предложен, разработан и апробирован новый **автоматный метод реализации проблемно-ориентированных языков**, в том числе визуальных, позволяющий по определению языка автоматически синтезировать его реализацию при определенных ограничениях
- Предложен и реализован **язык исполняемых программных спецификаций**, позволяющий описывать модели формализуемых предметных областей, ставить на них вычислительные задачи и синтезировать программы решения задач на основе логического вывода в специальном классе исчислений

Избранные публикации по теме диссертации (всего 51 публикация)

1. Новиков Ф.А. Программа для составления программ методом пошагового уточнения. — Алгоритмы небесной механики (материалы математического обеспечения ЭВМ), № 26, Ленинград, 1979 г., 16 стр.
2. Бабаев И.О., Новиков Ф.А., Петрушина Т.И. Язык Декарт – входной язык системы СПОРА // Прикладная информатика, сборник статей под ред. В.М.Савинкова, выпуск I, М., "Финансы и статистика", 1981 г., с. 35–72
3. Krasinsky G.A., Novikov F.A., Skripnichenko V.I. Problem Oriented Language for Ephemeris Astronomy and its Realization in System ERA // Celestial Mechanics, 1989, v.45, pp.219-229
4. Новиков Ф.А. Архитектура системы ЭРА — табличный подход к обработке данных — Сообщения ИПА РАН, № 16, 1990, 32 стр.
5. Нецветаева Г.А., Новиков Ф.А., Парийская Е.Ю. Система автоматической верстки табличных изданий (СВИТА) — Сообщения ИПА РАН, №118, 1998, 60 стр.
6. Новиков Ф.А., Яценко А.Д. Microsoft Office 97 в целом — СПб, BHV-Санкт-Петербург, 1999, 624 стр.
7. Новиков Ф.А. Визуальное конструирование программ // Информационно-управляющие системы, №6, 2005, с. 9–22
8. Новиков Ф.А., Степанян К.Б. Язык описания диаграмм // Информационно-управляющие системы, №4, 2007, с. 28–36
9. Клебан В.О., Новиков Ф.А. Применение конечных автоматов в документообороте // Научно-технический Вестник СПбГУ ИТМО Выпуск 53, Автоматное Программирование, 2008, с. 286–294
10. Новиков Ф.А., Новосельцев В.Б. Язык исполняемых программных спецификаций // Программирование, №1, 2010, 66-78
11. Новиков Ф.А. Иванов Д.Ю. Моделирование на UML. Теория, практика, видеокурс. — Санкт-Петербург, Наука и Техника, 2010, 640 стр.
12. Новиков Ф.А., Тихонова У.Н. Автоматный метод определения проблемно-ориентированных языков (Часть 1, 2, 3) // Информационно-управляющие системы, №6, 2009, с.34–40 , №2, 2010, с.31–37 , №3, 2010, с.29–37