ПРИМЕНЕНИЕ КОНТЕКСТНОЙ ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ ПРИ СОЗДАНИИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ СИСТЕМ УПРАВЛЕНИЯ КРУПНОМАСШТАБНЫМИ ПРОИЗВОДСТВАМИ

В.С. Выхованец

Институт проблем управления РАН

http://valery.vykhovanets.ru

1-3 октября 2008 г.

Предмет, объект и метод

Предмет – крупномасштабное производство как социальная система, адекватная модель которой не может быть получена в рамках одной или даже нескольких формальных теорий.

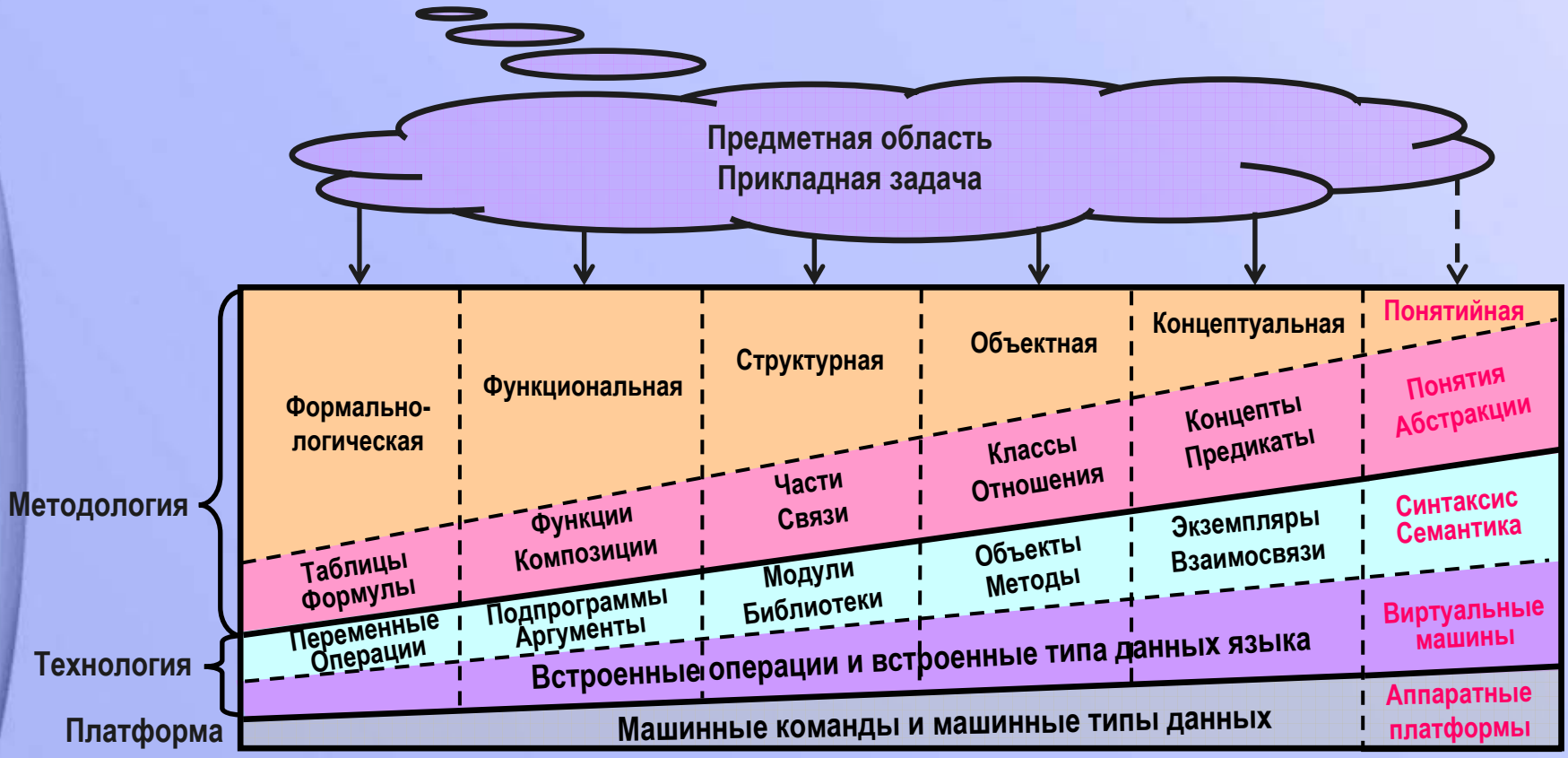
Объект – крупномасштабная система как совокупность многоуровневых иерархически организованных моделей различных типов.

Метод – иерархическая и многоаспектная декомпозиция предметной области в соответствии с накопленными о ней знаниями.

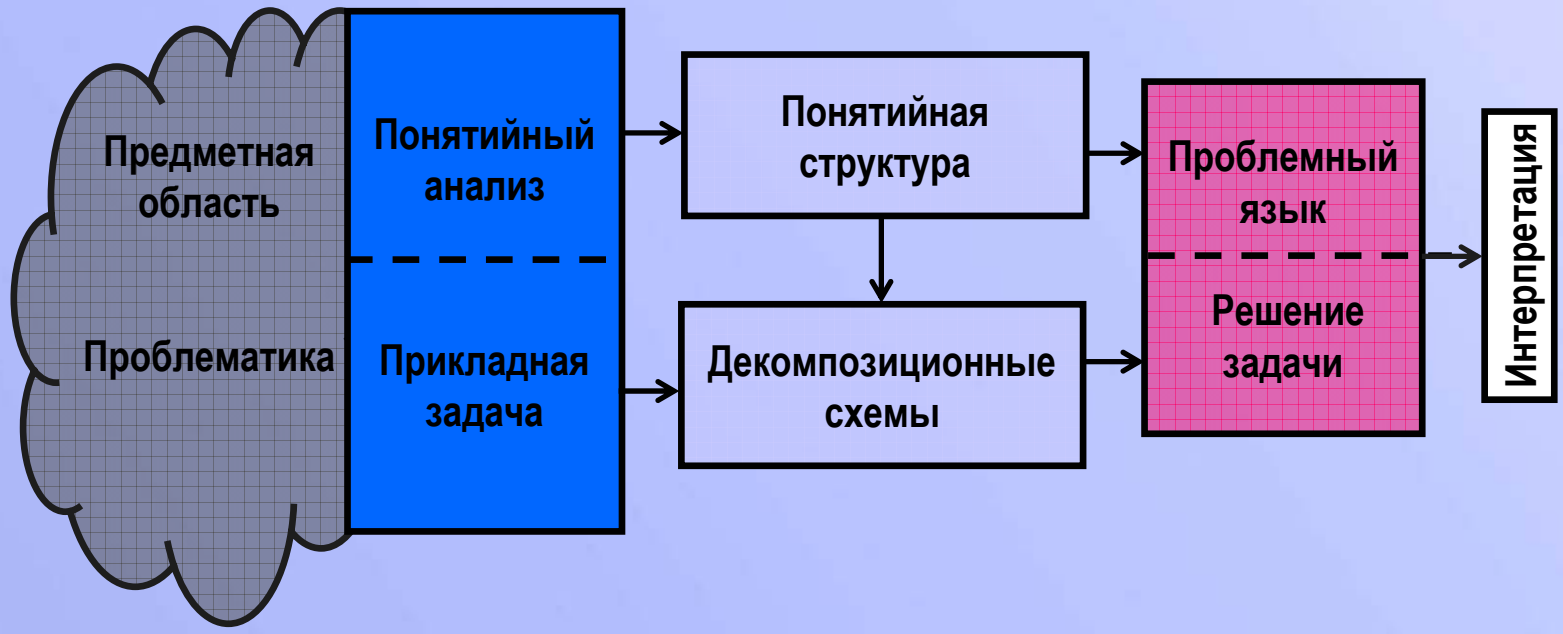
Проблематика

- Целенаправленность элементов
- Многоаспектный анализ
- Иерархическая декомпозиция
- Стыковка моделей различных типов
- Сложность верификации моделей
- Низкоуровневый формализм
- Семантический разрыв
- Трудности доработки

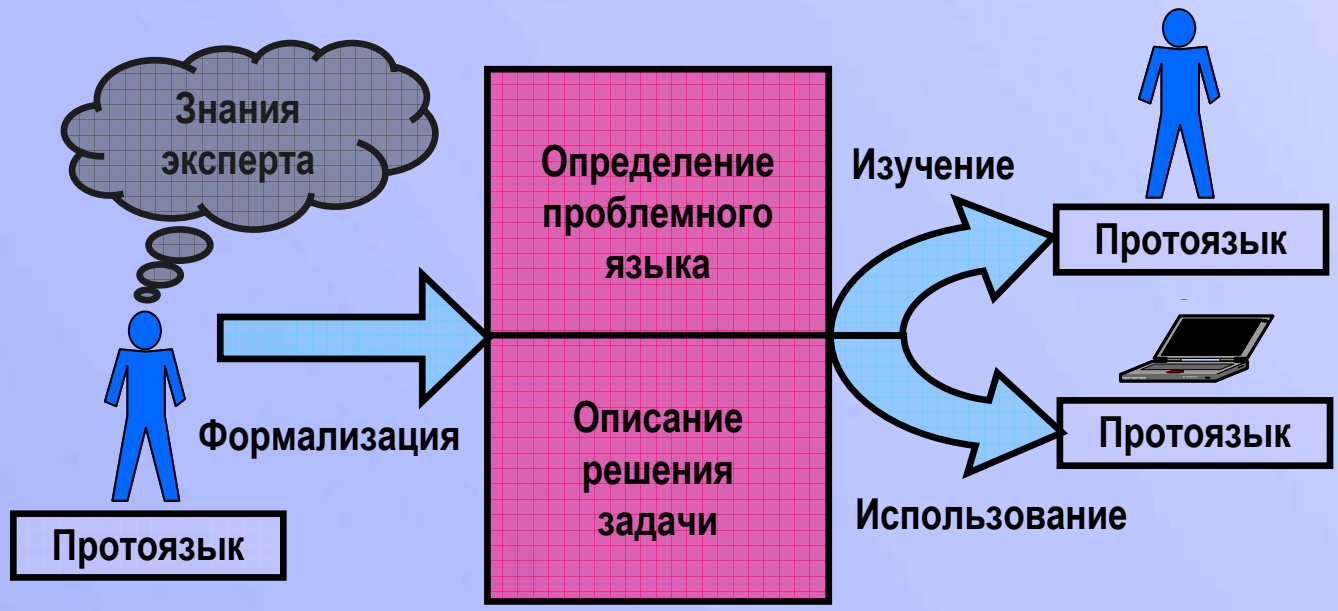
Суть подхода



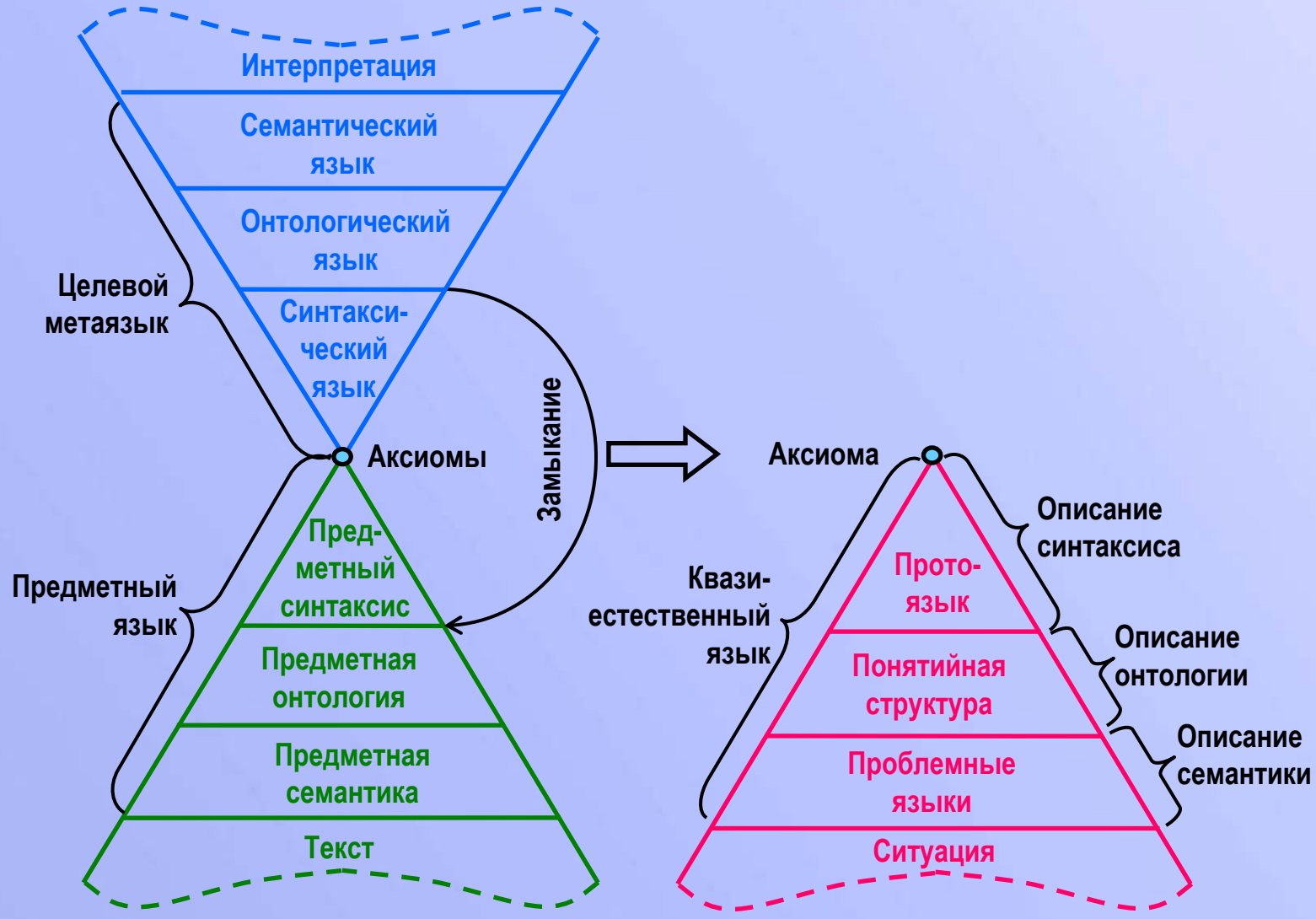
Методология



Технология


Знания эксперта
Протоязык
Формализация
Определение проблемного языка
Описание решения задачи
Изучение
Использование
Протоязык
Протоязык

Квазиестественный язык



Протоязык

model → essences [model]
essences → '(' ')' ***notion*** '(' ')' [intension]
intension → sentence [intension]
sentence → syntax '{' '}'
syntax → item [syntax]
item → **notion** | term
term → '"' [***terms***] '"'

$G(L) =< T, N, P, I >$

$G(L) =< \{0,1\}, \{I, J\}, P, I >$

$P = \{I \to 0 \mid J, J \to 1 \mid J0 \mid J1\}$

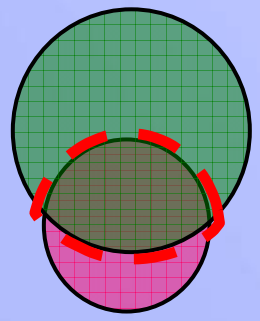
$I \to 0,$

$I \to J \to 1,$

$I \to J \to J0 \to 10.$

() ***J*** ()
'1' { }
J '0' { }
J '1' { }
() ***I*** ()
J { }
'0' { }

Абстракции



$$\begin{cases} \operatorname{shm} C_G = \bigcap_{j=0}^{m-1} \operatorname{shm} C_j; \\ \operatorname{int} C_G \supseteq \bigcup_{j=0}^{m-1} \operatorname{int} C_j; \\ \operatorname{ext} C_G \supseteq \bigcup_{j=0}^{m-1} \operatorname{ext} C_j. \end{cases}$$

3

Настольный

Сервер

Переносной

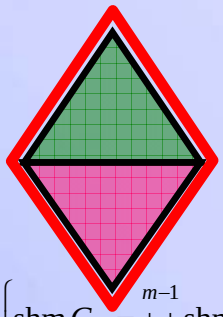
Компьютер

Клавиатура

Интерфейс

Клавиши

Корпус



$$\begin{cases} \operatorname{shm} C_A = \bigcup_{j=0}^{m-1} \operatorname{shm} C_j; \\ \operatorname{int} C_A = \times_{j=0}^{m-1} \operatorname{int} C_j; \\ \operatorname{ext} C_A = \times_{j=0}^{m-1} \operatorname{ext} C_j. \end{cases}$$

Обобщение

Агрегация

4

2

Ассоциация

Типизация

$$\begin{cases} \operatorname{shm} C_B \subseteq \bigcup_{j=0}^{m-1} \operatorname{shm} C_j; \\ \operatorname{int} C_B \subseteq \times_{j=0}^{m-1} \operatorname{int} C_j; \\ \operatorname{ext} C_B \subseteq \times_{j=0}^{m-1} \operatorname{ext} C_j; \\ \operatorname{link} C_B \subseteq \operatorname{shm} C_B. \end{cases}$$

Компьютер

Блок

Клавиатура

Мышь

Дисплей

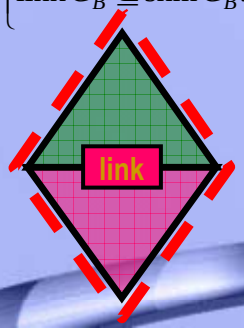
Мышь

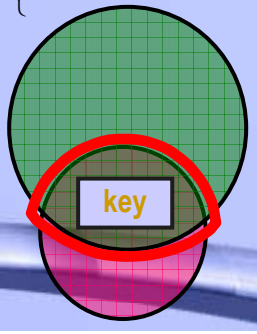
Оптическая

Лазерная

Беспровод.

$$\begin{cases} \operatorname{shm} C_T = \bigcap_{j=0}^{m-1} \operatorname{shm} C_j, \\ \operatorname{int} C_T = \bigcup_{j=0}^{m-1} \operatorname{int} C_j; \\ \operatorname{ext} C_T = \bigcup_{j=0}^{m-1} \operatorname{ext} C_j; \\ \operatorname{key} C_T \subseteq \operatorname{shm} C_T. \end{cases}$$

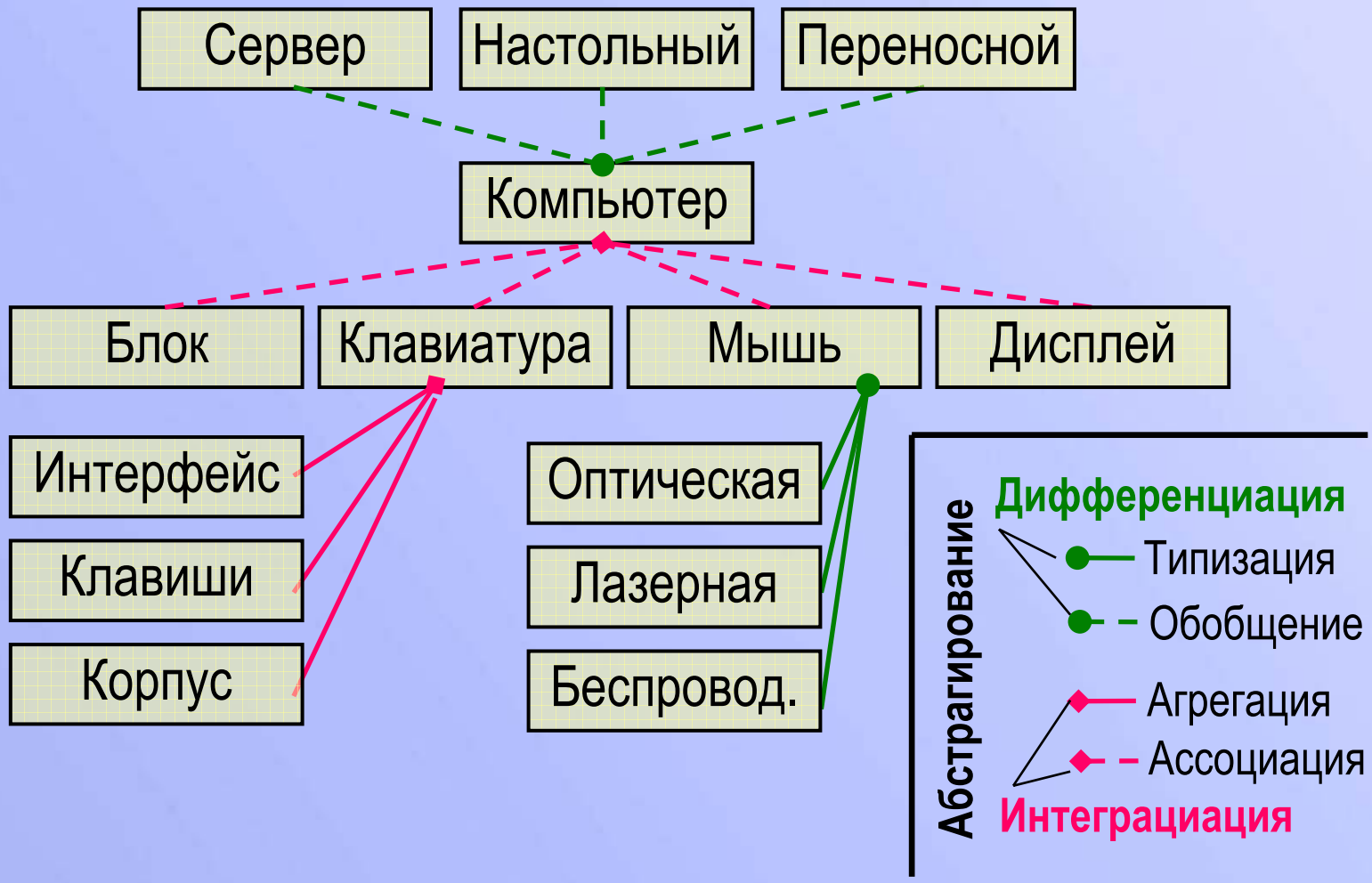




1

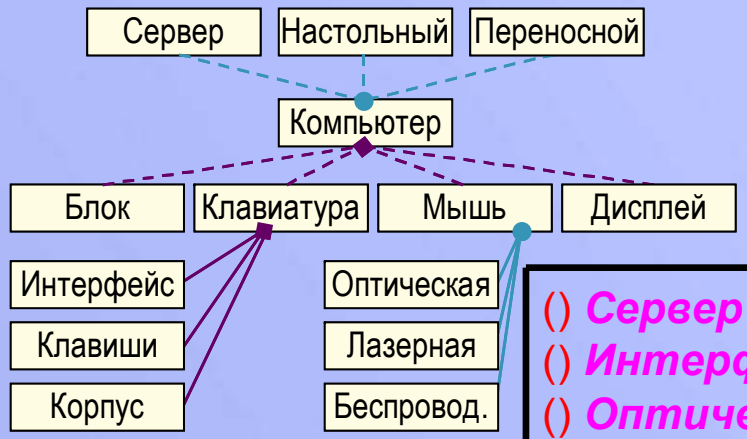
1-3 октября 2008 г.

Понятийная структура



Онтология

model → essences [model]
essences → differenciation ***notion***
integration [intension]
differenciation → '(' [notions] ')'
integration → '(' [notions] ')'
notions → **notion** [notions]



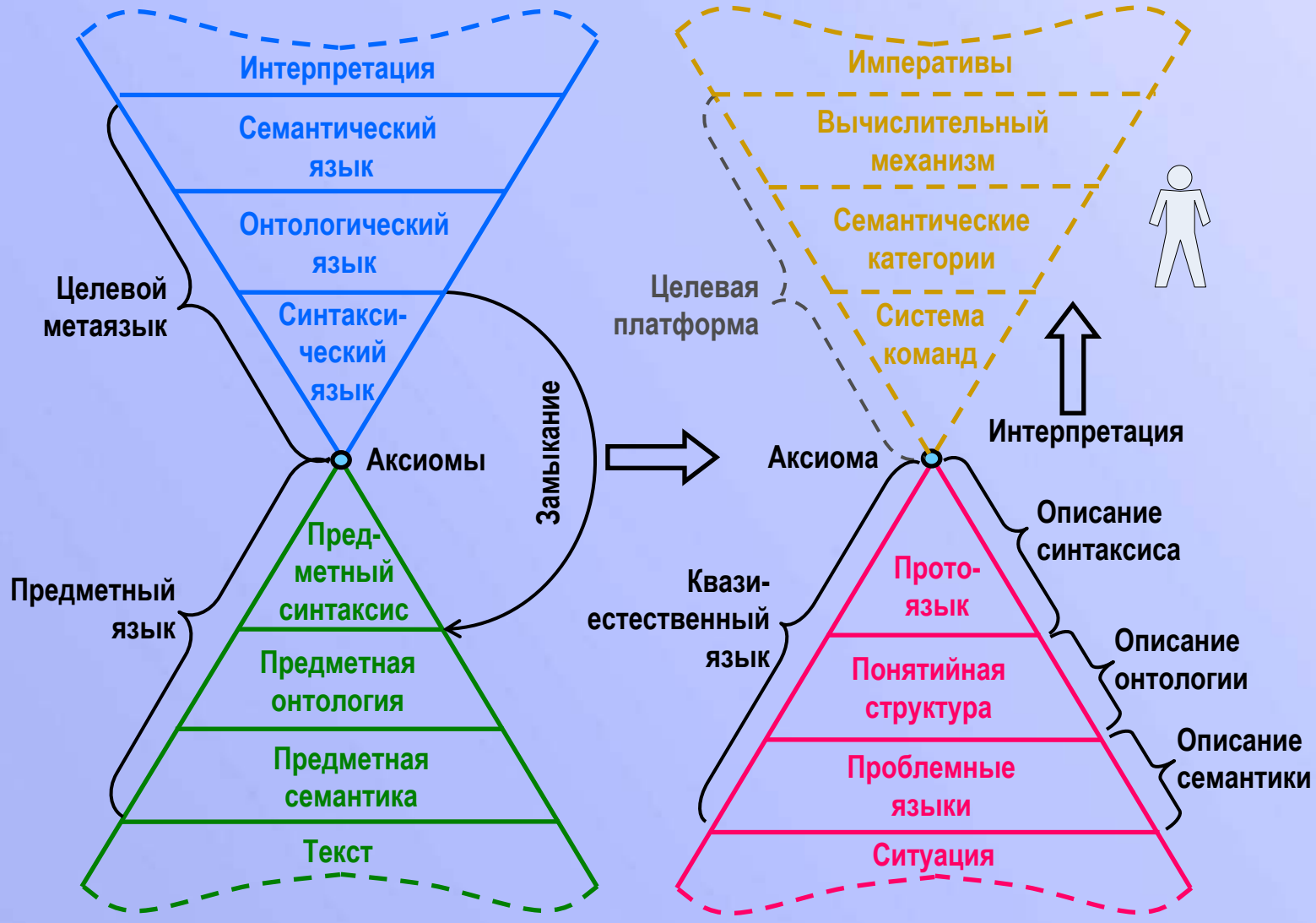
() ***Сервер*** () () ***Настольный*** () () ***Переносной*** ()
() ***Интерфейс*** () () ***Корпус*** () () ***Клавиши*** ()
() ***Оптическая*** () () ***Лазерная*** () () ***Беспроводная*** ()
() ***Блок*** () () ***Дисплей*** ()
(**Оптическая Лазерная Беспроводная**) ***Мышь*** ()
() ***Клавиатура*** (**Интерфейс Клавиши Корпус**)
(**Сервер Настольный Переносной**) ***Компьютер***
(**Блок Клавитура Мышь Дисплей**)

Проблемные языки

cognition → model [situation] [cognition]
model → essences [model]
essences → differenciation ***notion***
integration [intension]
intension → sentence [intension]
sentence → syntax semantic
syntax → item [syntax]
item → **notion** | '"' [***terms***] '"'
semantic → pragmatic [semantic]
pragmatic → [***aspect***] '**{**' [text] '**}**'
text → phrase [text]
phrase → **terms** | [**aspect**] '**{**' text '**}**'
situation → [**aspect**] '**<**' [text] '**>**'

```
() Boolean ()
    'false' {…}
    'true' {…}
    "[A-Za-z][A-Za-z0-9]*" {…}
    '(' Boolean ')' { }
    'not' Boolean {…}
    Boolean 'and' Boolean {…}
    Boolean [a] 'or' Boolean [b]
        { not (not a and not b) }
< (not x or y) and z >
fuzzy < not (x or y) and z >
```

Интерпретация



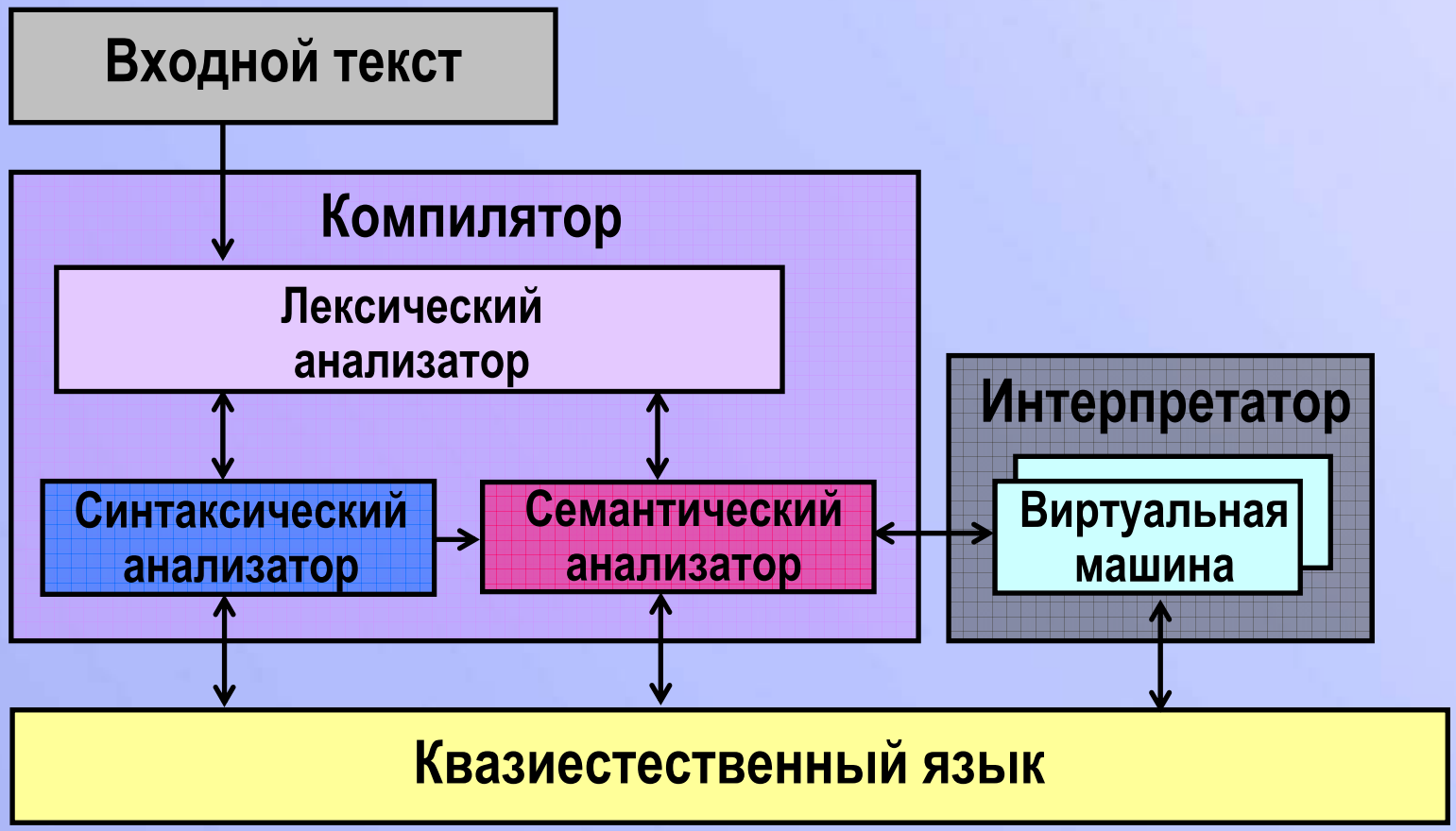
Аксиома

«Пустые» квадратные скобки реализуют запись в область императива (кода) того значения, которое задается текущим элементом предложения

```
() ()
    '#' "[0-9A-F][0-9A-F]" [ ] { }
() Boolean ()
    'not' Boolean
        { #58 #F7 #D0 #50 }
    Boolean 'or' Boolean
        { #58 #5A #0B #C2 #50 }
    Boolean [a] 'imp' [b] Boolean
        { not a or b }
```

```
() ()
    `' "[0-9A-Za-z\s.,]+" [ ] `' { }
() Boolean ()
    'not' Boolean
        { `pop eax` `not eax`
          `push eax` }
    Boolean 'or' Boolean
        { `pop eax` `pop edx`
          `or eax, edx` `push eax` }
    Boolean [a] 'imp' [b] Boolean
        { not a or b }
```

Система



Проблемные языки:
- определяемые;
- дополняемые;
- библиотечные;
- общедоступные.

Интерпретаторы:
- контроллеры, процессоры;
- абстрактные машины;
- операционные системы;
- системы программирования.

Пример

```
( ) Движение ( )
        "[Дд]вижение" {…}
( ) Дверь ( )
        "[Дд]верь" {…}
( ) Этаж ( )
        'этаж' "[0-9]" {…}
        'этаж' 'вызова' "|вверх|вниз" {…}
        'этаж' 'лифта' {…}
( ) Вызов (Этаж Движение)
        'вызов' {…}
( ) Лифт (Этаж Движение Дверь)
        'лифт' {…}
( ) Метка ( )
        "[А-Яа-я][А-Яа-я0-9]*" {…}
( ) Переход ( )
        Метка {…}
( ) Логическое ( )
        Этаж "выше|ниже|равен" Этаж {…
        Вызов "вверх|вниз|нет" {…}
        Дверь "открыта|закрыта" {…}
        'не ' Логическое { }
( ) ( )
        '. ' {…}
        Метка ':' {…}
        Движение "вверх|вниз|останов" {…}
        Дверь "открыть|закрыть" {…}
        'Если ' Логическое ', ' 'то ' Переход {…}
        'Если ' Логическое ', ' 'то ' Переход ', '
              'иначе ' Переход {…}
```

Ожидание:
    Если вызов нет, то Ожидание.
    Если этаж вызова равен этаж 2,
        то Открыть, иначе Решение.

Открыть:
    Дверь открыть.
    Если дверь открыта, то Решение,
        иначе Открыть.

Закрыть:
    Дверь закрыть.
    Если дверь закрыта, то Решение,
        иначе Закрыть.

Решение:
    Если лифт вверх и вызов вверх, то Вверх.
    Если лифт вниз и Вызов вниз, то Вниз.
    Если не вызов вверх и вызов вниз, то Вниз.
    Если не вызов вниз и вызов вверх, то Вверх.
    Если вызов нет и этаж лифта выше этаж 2,
        то Вниз.
    Если вызов нет и этаж лифта ниже этаж 2,
        то Вверх, иначе Ожидание.

Вверх:
    Движение вверх.
    Если этаж лифта равен этаж вызова вверх,
        то Открыть, иначе Вверх.

Вниз:
    Движение вниз.
    Если этаж лифта равен этаж вызова вниз,
        то Открыть, иначе Вниз.

Эффективность