СОЗДАНИЕ ПРЕДМЕТНО-ОРИЕНТИРОВАННОГО ЯЗЫКА ДЛЯ РАЗРАБОТКИ СЕРВЕРНОГО ПО

Асеева Серафима Олеговна, 17.Б10-мм
Науч. руководитель д.ф.-м.н., проф. А.Н. Терехов
Консультант Р.С. Ефремов

СХД и управляющее ПО

- Mainframe Enablers
- HPE Host Explorer
- Nutanix

Проблемы существующего решения

- Низкий уровень абстракции
- Императивный подход
- Отсутствие стандарта разработки

Варианты решения

- Разработка библиотеки на высокоуровневом языке

✓ Разработка предметно-ориентированного языка

Постановка задачи

1. Разработать декларативный язык программирования
2. Протестировать работу транслятора с помощью эмулятора СХД
3. Провести тестирование среди потенциальных пользователей

Примеры декларативных DSL

- SQL
- OCL
- Языки описания грамматик
- и т.д…

Синтаксис

```
<syscall description> ::= "syscall" "{" <fields list> "}" [<tag list>]
<fields list> ::= <none> | <fields list> <field>
<field> ::= <identifier> "["<size>"]" ["=" <hexadecimal number>] ";" [<tag list>]
<tag list> ::= <none> | <tag list> <tag>
<tag> ::= "#"<identifier>
<size> ::= <decimal number>

<command description> ::= "command" <identifier> "(" <parameter list> ")" "{"<rules list>"}"
<parameter list> ::= <none> | <parameter list>, <identifier>
<error description> ::= "error" <identifier> "{" <rules_list> "}"

<rules list> ::= <none> | <rules list> <rule>
<rule> ::= "DO" <tag> [":" <field tag assignments>] [", <errors list>] | PRINT <message>;
<errors list> ::= <errors list> ", ERROR =" <return code>, <identifier>
<return code> ::= <hexadecimal number> | DEFAULT
<field tag assignments> ::= <field tag assignments> "," <assignment>
<assignment> ::= <tag> "=" <hexadecimal number>
<message> ::= '"'<string of symbols>'"'

<comment> ::= "//" <string of symbols>

<letter> ::= [a-z,A-Z]
<digit> ::= [0-9]
<a-f letter> ::= [a-f, A-F]
<identifier> ::= "_" | <letter> | <identifier>"_" | <identifier><letter> | <identifier><digit>
<decimal number> ::= <digit> | <decimal number><digit>
<hexadecimal number> ::= "0x"<a-f letter> | "0x"<digit> | <hexadecimal number><a-f letter> | <hexadecimal number><digit>
<string of symbols> ::= "\n" | [-128 - 127]<string of symbols>
```

Пример

```
command snap_volume (src, tgt) {
        DO run_parallel_copy_validation: SRC = src, TGT = tgt,
                        ERR = (auto_PC_flag = 1),auto_PC_error, ERR = DEFAULT,PC_error;
        DO copy: SRC = src, TGT = tgt;
}
```

```
device_descriptor {
int cuu_number;
int sym_number;
char rdf_group_number;
char r2_rdf_group_number;
int rdf_group_number_extended;
int r2_rdf_group_number_extended;
char flags;
char microcode_version;
}
...
void snap_volume (device_descriptor src, device_descriptor tgt) {
    if (src.rdf_group_number_extended == tgt.rdf_group_number_extended){
        rc = run_parallel_copy_validation();
        if (rc==0)
            call_copy(src, tgt)
        else{
            if (auto_PC_stop){
                stop_PC();
                call_copy(src, tgt); }
            else
                printf("you have to remove parallel clone from the devices before copy")
        }
    }
}
```

```
error PC_error {
        PRINT "you have to remove parallel clone from the devices before copy";
}
error auto_PC_error {
        DO pc_stop: SRC = src, TGT = tgt;
        DO copy: SRC = src, TGT = tgt;
}
```

```
syscall sysc_pc_validation {
header[8] = 29000000;
cmd[4] = 0002;
auto_PC_flag[1] = 1;
...
source[4]; #SRC
target[4]; #TGT
...
} #run_parallel_copy_validation

syscall pc_stop_01 {
...
} #pc_stop
syscall pc_stop_02 {
...
} #pc_stop
...
syscall sysc_copy {
...
} #copy
```

Диаграмма последовательности для разработчика

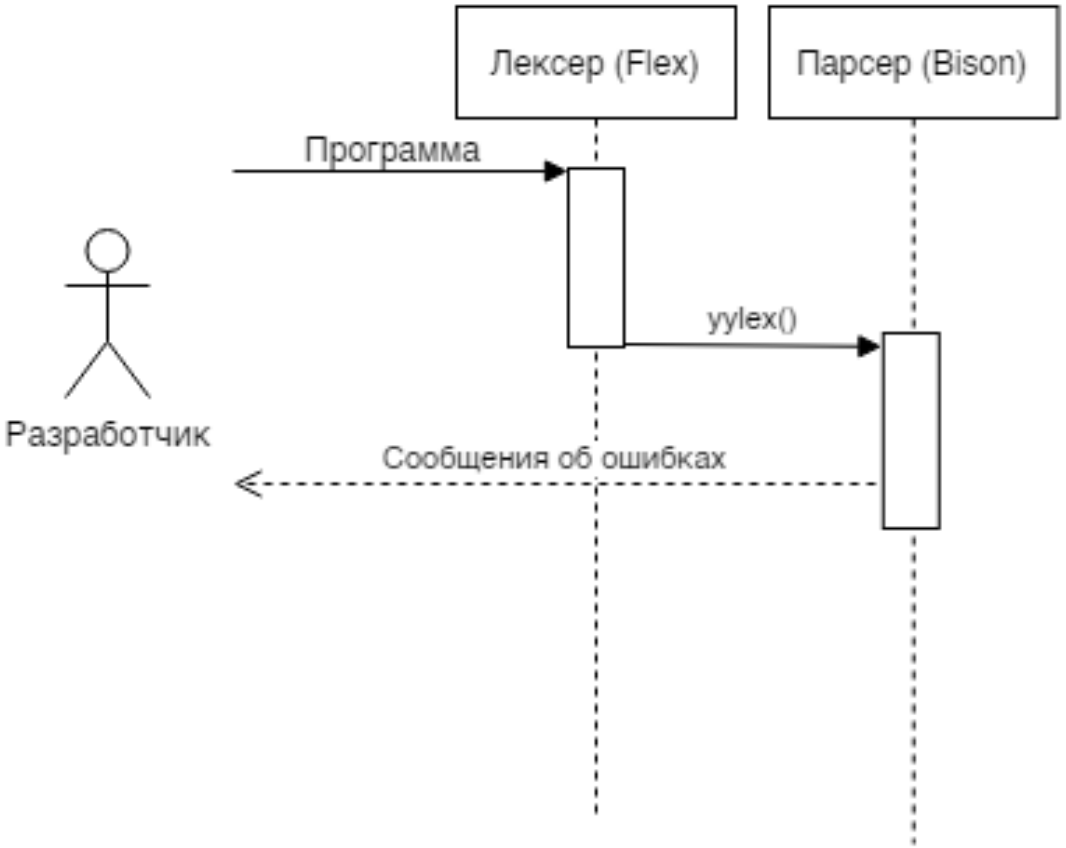
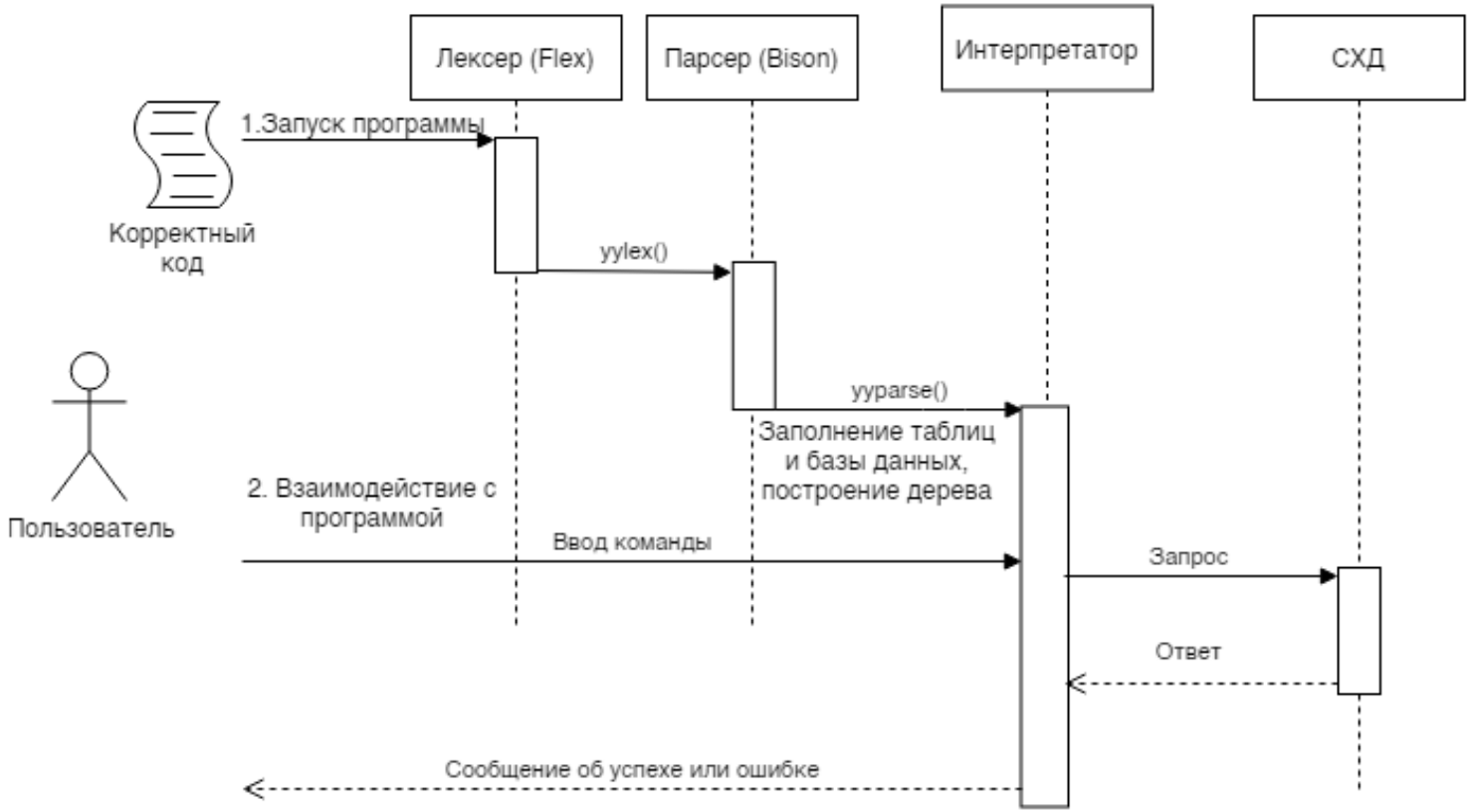


Диаграмма последовательности для пользователя



Итоги

- Описан синтаксис и семантика языка
- Реализован транслятор языка
  - Операции с базой данных и таблицами идентификаторов
  - Парсер
  - Интерпретатор
- Произведено юнит-тестирование
- Планируется протестировать производительность двух решений с помощью эмулятора, а также получить оценку потенциальных пользователей