

РАЗРАБОТКА И РЕАЛИЗАЦИЯ ЯЗЫКА ОПИСАНИЯ СЦЕНАРИЕВ ТЕСТИРОВАНИЯ АВТОМОБИЛЬНЫХ СИСТЕМ

выполнил
студент гр. 43504/6

А. А. Спасеева

руководитель
к.т.н., доцент

А. В. Самочадин

ВВЕДЕНИЕ

- В процессе написания требования для разрабатываемого ПО аналитики должны описать тестовые сценарии таким образом, чтобы их смог понять и разработчик, и тестировщик, и заказчик.
- Зачастую заказчики и аналитики не имеют достаточных знаний в языках программирования общего назначения.
- В данный момент производители АС пытаются найти решение, предоставляющее возможность описывать тестовые сценарии в форме, оговариваемой внутри команды.

ФОРМУЛИРОВКА ЗАДАЧИ

Цель работы: разработка и реализация проблемно-ориентированного языка для функционального тестирования автомобильных систем.

Для достижения цели необходимо решить задачи:

- Разработать язык, позволяющий пользователям описывать инструкции для тестовых сценариев в форме, оговариваемой внутри бизнес-команды.
- Реализовать и протестировать интерпретатор, предоставляющий пользователю возможность работы с автомобильными сетями.

ОБЗОР СУЩЕСТВУЮЩИХ ИНСТРУМЕНТОВ ТЕСТИРОВАНИЯ

Инструмент Критерии сравнения	CAPL	CCDL	Cucumber
Возможность описать несколько тестовых сценариев в документе	+	+	+
Поддержка возможности работы с шинами АС	+	+	-
Циклическое выполнение тестовых шагов	+	+	-
Возможность описывать инструкции тестовых сценариев в свободной форме	-	-	+

ПОСТАНОВКА ЗАДАЧИ

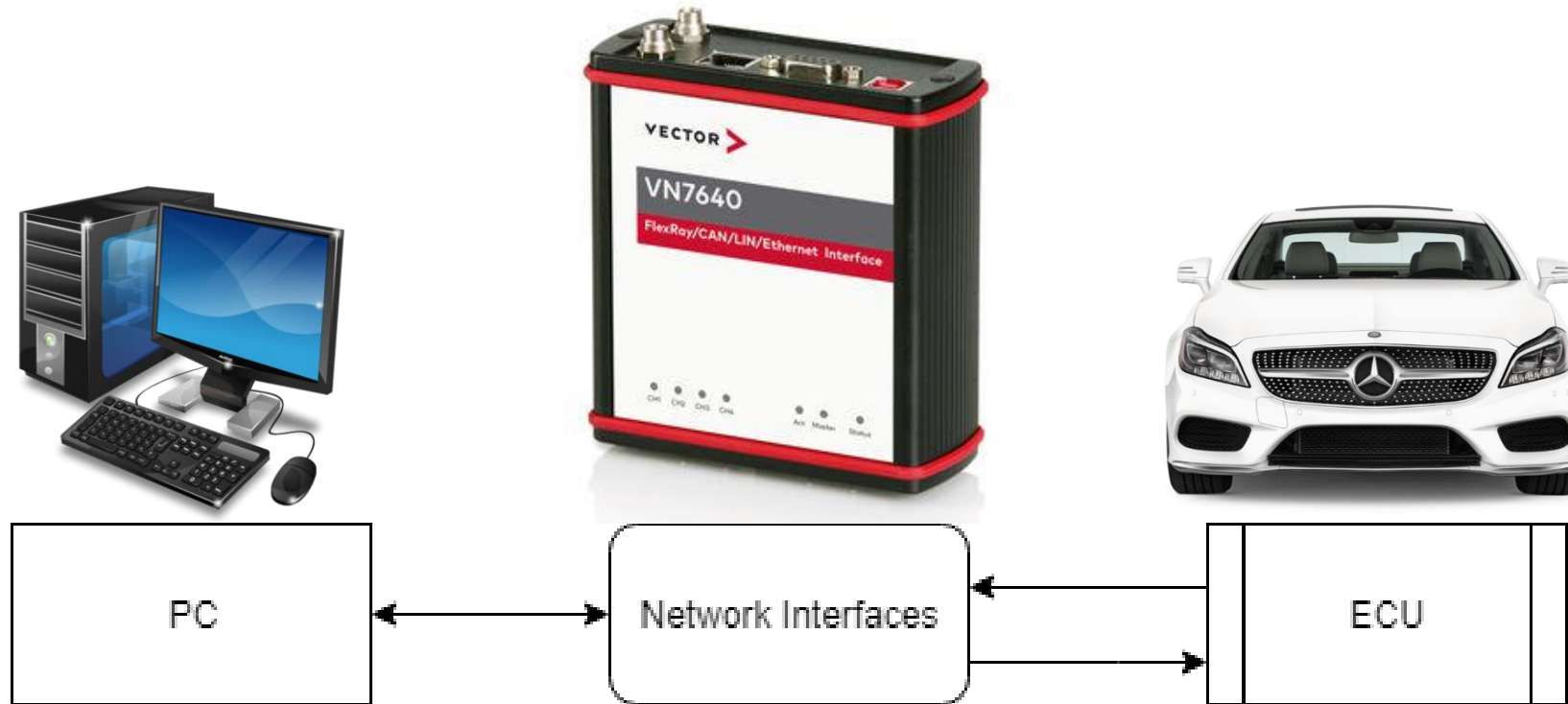
Требования к работе:

- Разработать язык
 - Язык должен позволять пользователям описывать инструкции для тестовых сценариев в форме, оговариваемой внутри бизнес-команды.
 - Синтаксис языка должен позволять описывать несколько тестовых сценариев в документе.
- Реализовать интерпретатор
 - Тестовые сценарии должны выполняться в той очередности, в которой описаны в документе.
 - Необходимо предоставить пользователю возможность работы с шинами **CAN** и **FlexRay**.
 - Результаты выполнения тестов должны сохраняться в файл.

Качество решения, представленного в работе должно быть проверено следующим образом:

- Провести опрос потенциальных пользователей для выявления степени удобства использования языка.
- Выполнить тестирование реализации языка.

АППАРАТНАЯ ПЛАТФОРМА



РАЗРАБОТКА ЯЗЫКА

@TestCase test NVRAM functionality of did 0xF101 (Req_NVRAM1)

@Set ignition on

@Send request 2E F1 01 00 01 A5

@Receive response 6E F1 01

@Set ignition off

@Set ignition on

@Repeat[2][

@Send request 22 F1 01

@Pause [100] ms

@Receive response 62 F1 01 00 01 A5

]

@Set ignition off

РАЗРАБОТКА ЯЗЫКА

@TestCase test NVRAM functionality of did 0xF101 (Req_NVRAM1)

@Set ignition on

@Send request [2E F1 01 00 01 A5]

@Receive response [6E F1 01]

@Set ignition off

@Set ignition on

@Repeat[2][

@Send request [22 F1 01]

@Pause [100] ms

@Receive response [62 F1 01 00 01 A5]

]

@Set ignition off

@Send request 2E F1 01 00 01 A5

@Send request 22 F1 01

@Receive response 6E F1 01

@Receive response 62 F1 01 00 01 A5



@Send request [data]

@Receive response [data]

РАЗРАБОТКА ЯЗЫКА

@TestCase test NVRAM functionality of did 0xF101 (Req_NVRAM1)

@Set ignition on

@Send request [2E F1 01 00 01 A5]

@Receive response [6E F1 01]

@Set ignition off

@Set ignition on

@Repeat[2][

@Send request [22 F1 01]

@Pause [100] ms

@Receive response [62 F1 01 00 01 A5]

]

@Set ignition off

ANTLR4

```
instruction
    : Send instrText      #send
    | Receive instrText   #receive
    | Set instrText        #set
    | Check instrText     #check
    | Pause time TEXT?    #pause
    ;

repeatPart
    : Repeat time LBRACKET repeatScen=scenario RBRACKET
    ;

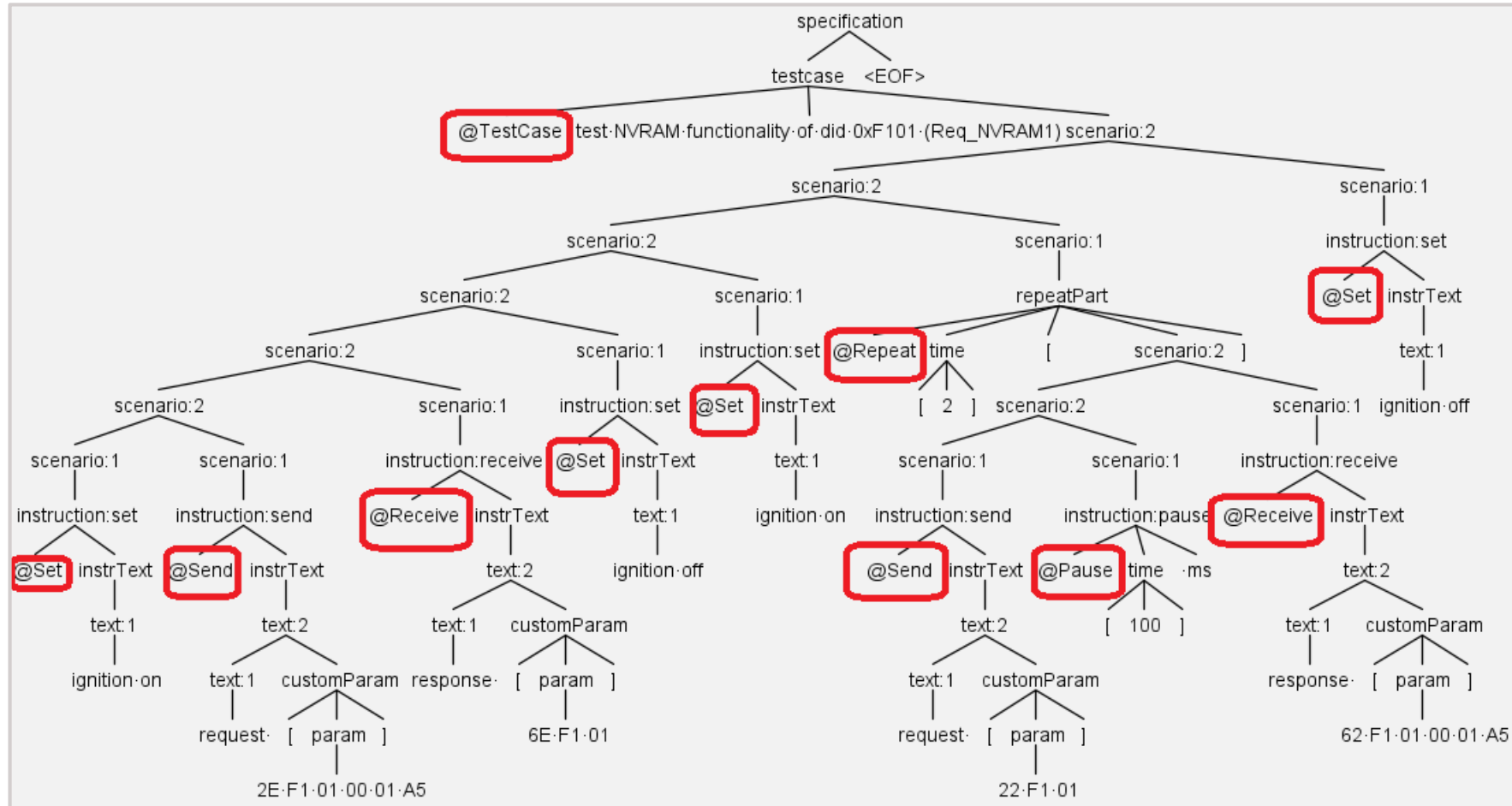
text
    : TEXT
    | left = text customParam (right = text)?
    ;
```

Для разработки и реализации языка был выбран генератор нисходящих анализаторов для формальных языков ANTLR4. Он преобразует контекстно-свободную грамматику в РБНФ в программу на Java.

```
Send
    : DOG [Ss][e][n][d]
    ;

Receive
    : DOG [Rr][e][c][e][i][v][e]
    ;
```

AST



РЕАЛИЗАЦИЯ ЯЗЫКА

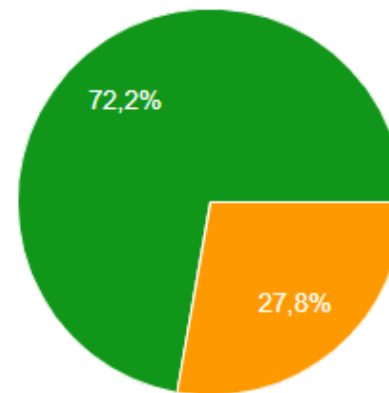
```
public class CommandLib extends XLCommandLib {  
  
    @Send(text = "request [param1]")  
    public void sendRequest(String param1) {  
        |    xlFrTransmit(HexUtil.fromStringtoHexArray(param1));  
    }  
  
    @Receive(text = "response [param1]")  
    public void receiveResponse(String param1) {  
        |    xlFrReceive(HexUtil.fromStringtoHexArray(param1));  
    }  
}
```

```
public class SpecBddListener extends BddParserBaseListener {  
  
    @Override  
    public void enterSend(BddParser.SendContext ctx) {  
        |    findMethodByInstructionContext(KeyWord.SEND, ctx);  
    }  
}
```

Для симуляции автомобильных сетей используется универсальный программный интерфейс **XL-Driver-Library**, позволяющий получить доступ к интерфейсам аппаратных средств Vector.

ТЕСТИРОВАНИЕ

- ✓ ParserTest (main.tests)
 - ✓ testPauseGoodValPauseNullVal
 - ✓ testSendEmptyInstructionEOF
 - ✓ testInstructionIsNotDeclared
 - ✓ testSendEmptyInstruction
 - ✓ testRepeatNoLBracket
 - ✓ testPauseNullValule
 - ✓ testRepeatNoTimeToRepeatWithBrackets
 - ✓ testSetEmptyInstruction
 - ✓ testTwoTestCaseNoDescr
 - ✓ testRepeatTimeToRepeatWithoutBrackets
 - ✓ testRepeatNoRBracket
 - ✓ testRepeatWithoutSteps
 - ✓ testPauseValule
 - ✓ testNoTestCaseKeyword



- спецификация непонятна
- спецификация понятна, но я бы не стал(а) описывать подобным образом
- спецификация понятна, но я бы изменил(а) некоторые моменты
- спецификация понятна, по возможности могла бы использоваться в работе

АПРОБАЦИЯ

@TestCase test NVRAM functionality of did 0xF101 (Req_NVRAM1)

@Set ignition on

@Send request [2E F1 01 00 01 A5]

@Receive response [6E F1 01]

@Set ignition off

@Set ignition on

@Repeat[2][

@Send request [22 F1 01]

@Pause [100] ms

@Receive response [62 F1 01 00 01 A5]

]

@Set ignition off

TestCase test NVRAM functionality of did 0xF101 (Req_NVRAM1)

[6E F1 01] SUCCESS

[62 F1 01 00 01 A5] FAILED: [7F 22 31]

[62 F1 01 00 01 A5] FAILED: [7F 22 31]

ЗАКЛЮЧЕНИЕ

- Разработан язык
 - Язык позволяет пользователям описывать инструкции для тестовых сценариев в форме, оговариваемой внутри бизнес-команды.
 - Синтаксис языка позволяет описывать несколько тестовых сценариев в документе.
- Реализован интерпретатор
 - Тестовые сценарии выполняются в той очередности, в которой описаны в документе.
 - Реализована возможность работы с шинами **CAN** и **FlexRay**.
 - Результаты выполнения тестов сохраняются в файл.
- Проверено качество решения
 - Выполнено модульное тестирование интерпретатора.
 - Проведен опрос среди потенциальных пользователей для выявления степени удобства использования языка.

**СПАСИБО ЗА
ВНИМАНИЕ!**