и показателей. При создании программной модели для решения задач защиты речевой информации целесообразно использовать программную среду ANSYS. ANSYS является универсальной программной системой конечно-элементного анализа, работающей на основе геометрического ядра Parasolid, которая может быть использована для решения широкого спектра задач, в том числе и задач акустики [1].

Для создания программной модели защиты речевой информации в среде ANSYS необходимо определение требуемого количества параметров и показателей, используемых для построения модели. К таким параметрам можно отнести размеры помещения, местоположение и размеры источника звука, плотность воздуха, тип среды и тип колебаний, скорость звука, частота звука и др. [2, 3]. Выбор параметров должен позволять модели определять оптимальные условия для предотвращения или минимизации утечки речевой информации за счет несанкционированного прослушивания или изменения речевого сообщения посредством модификации информации или индивидуальных особенностей говорящего. Разрабатываемая программная модель должна иметь возможность исследовать зависимость распространения речевой информации от различных физических характеристик акустической среды и их сочетаний, и уязвимости речевой информации при изменении какой-либо характеристики. Такие возможности программной модели обеспечиваются за счет определения общих принципов разбиения объектов на конечные элементы, поиска системы с оптимальными параметрами за счет моделирования с использованием различных вариантов разбиения объектов на конечные элементы, оценки результатов данного моделирования и выявления оптимального разбиения объектов на конечные элементы.

Возможности данной программы значительно упрощают процедуры создания требуемой модели и оценку результатов, позволяют использовать интерактивную графику для проверки геометрии модели. Вывод графической информации на экран способствует проведению контроля результатов проектных решений непосредственно в процессе работы. Применение данной программы позволяет избежать дорогостоящих и длительных циклов разработки и получить требуемые результаты в минимально короткие сроки.

Таким образом, разработка программной модели защиты речевой информации с помощью средств многоаспектного моделирования, реализованных в ANSYS, и оценка результатов моделирования на основании построенной модели позволит определить оптимальных методы защиты речевой информации и разработать рекомендации по защите речевой информации от утечки по акустическим каналам.

Список использованных источников:

1. Басов, К.А. ANSYS: справочник пользователя / К. А. Басов. – М.: ДМК Пресс, 2005. – 640 с.
2. Хорев, А. А. Методы защиты речевой информации и оценки их эффективности / А. А. Хорев, Ю. К. Макаров // Защита информации. Конфидент, 2001. – № 4. – С. 22-33.
3. Хорев, А. А. Оценка эффективности защиты информации от утечки по техническим каналам / А. А. Хорев // Специальная техника, 2006. – № 6. – С. 53-61.

ФУНКЦИОНАЛЬНОЕ ПРОГРАММИРОВАНИЕ В JVM

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Селивошко В. И., Шинкевич А. А.

Сечко Г. В. – канд. техн. наук, доцент

Рассматриваются возможности функционального программирования в рамках наиболее распространённых языков для JVM

1 Что такое JVM? Java Virtual Machine (*JVM*) — *виртуальная машина Java* — основная часть исполняющей системы Java, так называемой *Java Runtime Environment* (*JRE*). *JVM* интерпретирует *байт-код Java*, предварительно созданный из исходного текста *Java-программы компилятором Java* (*javac*). *JVM* может также использоваться для выполнения программ, написанных на других языках программирования (ЯП). Например, исходный код на языке *Ada* может быть откомпилирован в *байт-код Java*, который затем может выполниться с помощью *JVM*. *JVM* доступны для многих аппаратных и программных платформ, что позволяет описать *Java* как «скомпилировано однажды, запускается везде» (compile once, run anywhere) [1].

2 Краткий обзор доступных JVM ЯП. JVM, кроме непосредственно программ, написанных на Java, позволяет интерпретировать *байт-код Java*, полученный в процессе компиляции исходного кода одного из множества допустимых ЯП. Список наиболее часто используемых ЯП, доступных для запуска на *JVM* [2, 3]:

Clojure, функциональный диалект языка программирования Lisp;

Processing, объектно-ориентированный язык для создания изображений и анимации;

Groovy, объектно-ориентированный, скриптовый ЯП;

Scala, мультипарадигмальный ЯП, сочетающий в себе возможности как функционального, так и объектно-ориентированного программирования;

JRuby, реализации языка *Ruby*;

Jython, реализация языка *Python*;

Rhino, реализация языка *JavaScript*;

Kotlin, статически типизированный объектно-ориентированный ЯП компании JetBrains.

3 Функциональное программирование (ФП) в Java. ЯП Java является объектно-ориентированным и не содержит явных средств для ФП. Однако в нём существует поддержка внутренних анонимных классов, позволяющих создавать объекты функций, которые можно использовать схожим образом с функциями в языках ФП. Для того чтобы создать объект функции, необходимо сначала описать его интерфейс, обычно содержащий один метод, после чего можно создавать объекты функций, реализуя метод интерфейса в теле анонимного класса [2]. Ядром ФП является функция. Функции образуют строительные блоки для обработки данных и имеют дополнительные возможности, не встречающиеся в традиционных императивных языках. Функции высших порядков могут принимать другие функции в качестве аргументов или возвращать их в виде результатов. В языке Java подобные конструкции не допускаются. В качестве ближайшего аналога можно рассмотреть использование класса (обычно анонимного) в качестве "контейнера" для функции, которую необходимо выполнить. В Java не существует самостоятельных функций или методов, так что их невозможно возвращать из функций или передавать в качестве параметров [4].

В *Java* до седьмой версии включительно можно создать аналог *функции высшего порядка* при помощи *анонимного класса*, реализующего метод некоторого интерфейса. В [5] приведен пример использования такого класса. В [6] рассматривается использование интерфейса *Comparator* в качестве *функции высшего порядка.* В JDK (Java Development Kit) версии *8* появятся специальные языковые конструкции для работы с такого рода функциями. И, к примеру, анонимный класс, реализующий интерфейс *Comparator*, можно будет переписать в функциональном стиле, создав *лямбда-выражение* (*лямбда-выражения* могут реализовывать любой *функциональный интерфейс,* представляющий собой интерфейс с одним абстрактным методом):

```
Comparator<Integer> cmp = (x, y) -> (x < y) ? -1 : (x > y) ? 1: 0;                    (1)
```

Стоит заметить, что код стал более компактным и лучше читаем.

4 ФП в Clojure. Clojure — современный диалект Лиспа. Это ЯП общего назначения с поддержкой разработки в интерактивном режиме, поощряющий ФП и упрощающий поддержку многопоточности [9]. Синтаксис Clojure основан на S-выражениях, которые перед компиляцией транслируются синтаксическим анализатором в структуры данных. Синтаксический анализатор Clojure поддерживает, помимо обычных списков, синтаксис литералов для ассоциативных массивов, множеств и векторов, передавая затем все эти структуры данных компилятору. Иначе говоря, компилятор Clojure компилирует не только списковые структуры данных, но напрямую поддерживает все указанные типы. Clojure является расширением изначальной версии Lisp, и не предназначен для кода совместимого с другими диалектами Лиспа [9]. Особенности языка Clojure:

динамическая, интерактивная разработка в REPL-цикле;

функции как объекты первого класса с акцентом на рекурсию, а не на основанную на побочных эффектах итерацию;

«ленивые» последовательности;

обеспечение богатого набора неизменяемых, сохраняемых структур данных;

параллельное программирование с поддержкой транзакционной памяти, агентной системы и системы динамических переменных;

тесная интеграция с Java: за счёт компиляции в байткод JVM программы на Clojure легко переносятся в любую среду с JVM. Язык также обеспечивает ряд макросов, которые упрощают использование в нём существующих Java API. Структуры данных Clojure реализуют все стандартные интерфейсы Java, что делает легким запуск из Java программного кода написанного на Clojure [9].

5 ФП в Scala. Scala — мультипарадигмальный ЯП, спроектированный кратким и типобезопасным для простого и быстрого создания компонентного программного обеспечения, сочетающий возможности функционального и объектно-ориентированного программирования [7]. Ключевые аспекты языка:

Scala-программы во многом похожи на *Java-программы*, и могут свободно взаимодействовать с *Java-кодом*;

Scala включает единообразную объектную модель — в том смысле, что любое значение является объектом, а любая операция — вызовом метода;

Scala — это также функциональный язык в том смысле, что функции — это полноправные значения;

В *Scala* включены мощные и единообразные концепции абстракций как для типов, так и для значений;

Scala содержит гибкие симметричные конструкции примесей для композиции классов и *trait-ов (подобие интерфейсов в Java)*;

Scala позволяет производить декомпозицию объектов путем сравнения с образцом;

Образцы и выражения были обобщены для поддержки естественной обработки *XML-документов*;

Конструкции *Scala* позволяют легко выражать самостоятельные компоненты, использующие библиотеки *Scala*, не пользуясь специальными языковыми конструкциями;

Scala допускает внешние расширения компонентов с использованием видов (*views*);

В *Scala* есть шаблоны (*generics*) и шаблоны высших порядков(*generics of a higher kind*);

В *Scala* имеется поддержка структурных и экзистенциальных типов;

На текущий момент Scala реализована на платформах Java и .NET. Рассмотрим программу в функциональном стиле на языке *Scala* [8].

```
object Timer {
  def periodicCall(seconds: Int, callback: () => Unit): Unit = {
            while (true) {
  callback();
  Thread.sleep(seconds * 1000);                                                    (2)
}
```

```
}
def main(args: Array[String]): Unit = {
        periodicCall(1, () =>
                Console.println("Time flies… oh, you get the idea."));
}
}
```

Функция *main* в (2) передает произвольный блок кода в качестве параметра *oncePerSecond*, это выглядит как *лямбда-выражение* из *Lisp* или *Scheme*, что, само по себе, является разновидностью замыкания. Такая *анонимная функция* демонстрирует мощь отношения к функциям как к гражданам "первого сорта", позволяя обобщать код таким совершенно новым способом, не прибегая к механизму наследования. Не слишком сложно представить, как мог бы выглядеть *Java*-эквивалент рассмотренного выше кода и не слишком трудно признать, что версия *Scala* гораздо, гораздо короче и намного очевиднее.

Материалы данного доклада в расширенном виде апробированы студентами-вечерниками ИИТ специальности «Программное обеспечение информационных технологий» весной 2013 года как вспомогательные материалы по курсу «Функциональное программирование».

Список использованных источников

1. Виртуальная машина Java [Электронный ресурс] – 2013 – Режим доступа: http://ru.wikipedia.org/wiki/Java_Virtual_Machine. – Дата доступа: 20.03.2013.

2. Список языков программирования, доступных для JVM [Электронный ресурс] – 2013 – Режим доступа: http://en.wikipedia.org/wiki/List_of_JVM_languages. – Дата доступа: 20.03.2013.

3. Язык программирования Java [Электронный ресурс] – 2013 – Режим доступа: http://ru.wikipedia.org/wiki/Java. – Дата доступа: 20.03.2013.

4. Функциональное мышление: Часть 1. Разработка программ в функциональном стиле [Электронный ресурс] – 2012 – Режим доступа: http://www.ibm.com/developerworks/ru/library/j-ft1/index.html. – Дата доступа: 20.03.2013.

5. Функциональное программирование в Java [Электронный ресурс] – 2011 – Режим доступа: http://habrahabr.ru/post/122919/. – Дата доступа: 20.03.2013.

6. JSR 335 или lambda-выражения в JAVA 8 [Электронный ресурс] – 2012 – Режим доступа: http://habrahabr.ru/post/155191/. – Дата доступа: 20.03.2013.

7. Язык программирования Scala [Электронный ресурс] – 2013 – Режим доступа: http://ru.wikipedia.org/wiki/Scala_(язык_программирования). – Дата доступа: 20.03.2013.

8. Путеводитель по Scala для Java-разработчиков: Функциональное программирование вместо объектно-ориентированного [Электронный ресурс] – 2008 – Режим доступа: http://www.ibm.com/developerworks/ru/library/j-scala01228/. – Дата доступа: 20.03.2013.

9. Язык программирования Clojure [Электронный ресурс] – 2013 – Режим доступа: http://ru.wikipedia.org/wiki/Clojure. – Дата доступа: 12.04.2013.

МОДУЛЬ УПРАВЛЕНИЯ И ВИЗУАЛИЗАЦИИ СИСТЕМЫ МОДЕЛИРОВАНИЯ ДИНАМИКИ ЛЕТАТЕЛЬНЫХ ОБЪЕКТОВ

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Сенько Р. А.

Пачинин В. И. – зав. кафедрой ИСиТ ИИТ

Под воздействием интенсивного развития информационных технологий меняются принципы построения, характеристики, задачи и облик современных автоматизированных систем управления (АСУ), существенно возрастают их возможности. Основными направлениями совершенствования АСУ военного назначения в настоящее время являются создание новых современных ЭВМ и их функциональных подсистем, а также способов обработки и представления информации.

Разработанный программный модуль управления и визуализации (далее модуль управления) входит в состав пакета прикладных программ, обеспечивающих моделирование характеристик баллистических объектов [1].

Модуль управления представляет собой отдельное приложение, задачами которого являются:

- управление согласованной работой остальных модулей, входящих в состав пакета прикладных программ (модуль имитации характеристик баллистических объектов, модуль исходных данных);
- отображение с использованием трёхмерной графики траекторий полета объектов;
- отображение динамики полета летательного объекта в различных системах координат.

Компоненты архитектуры входящие в состав модуля управления приведены на рисунке 1.