DOI 10.37882/2223-2966.2025.02-2.29

РАЗРАБОТКА АВТОМАТИЗИРОВАННОЙ ИНФОРМАЦИОННОЙ СИСТЕМЫ АКУСТИЧЕСКИХ РАСЧЕТОВ И ИССЛЕДОВАНИЙ

DEVELOPMENT OF AN AUTOMATED INFORMATION SYSTEM FOR ACOUSTIC CALCULATIONS AND RESEARCH

O. Pleskacheva
B. Pruss
V. Romanov

Summary. The article describes the developed information system that allows performing acoustic calculations. The developed system has a user-friendly interface and the necessary functions that allow the user to obtain solutions both in numerical form and in the form of graphic images. The developed system is used to solve problems of construction acoustics studied by students within the framework of the course on the physical principles of construction acoustics. The results of using the information system show that it correctly and accurately performs calculations, and at the same time the system itself is convenient and easy to use.

Keywords: information system, acoustics, automation of calculations.

Плескачева Ольга Юрьевна
Кандидат педагогических наук, доцент, ФГБОУ ВО «Брянский государственный технический университет»
pleskacheva@inbox.ru
Прусс Борис Наумович
Кандидат технических наук, доцент, ФГБОУ ВО «Брянский государственный инженерно-технологический университет»
prussbor@gmail.com
Романов Виктор Александрович
Кандидат технических наук, доцент, ФГБОУ ВО «Брянский государственный инженерно-технологический университет»
vromanov62@mail.ru

Аннотация. В статье описана разработанная автоматизированная информационная система, позволяющая осуществлять акустические расчеты. Разработанная система имеет удобный интерфейс и необходимые функции, позволяющие пользователю получать решения как в числовом виде, так и в виде графических изображений. С помощью разработанной системы решаются задачи строительной акустики, изучаемые студентами в рамках курса физические основы строительной акустики. Результаты использования информационной системы показывают, что она корректно и точно выполняет расчеты и при этом сама система удобна и проста в обращении.

Ключевые слова: информационная система, акустика, автоматизация расчетов.

Акустика является областью физики, изучающей упругие колебания и волны, процесс их возбуждения и распространения, взаимодействие их с веществом и разнообразные применения. К прикладным областям акустики можно отнести строительную акустику, которая изучает вопросы снижения уровня шума в зданиях. Кроме основного пути борьбы с шумом, то есть снижения уровня шума в источнике, во многих случаях требуется использование других методов: звукоизоляции, звукопоглощения [1]. Поэтому проведение акустических расчетов и исследований является актуальной задачей, решение которой необходимо учитывать при проектировании и реконструкции жилых домов и инфраструктурных объектов в современном городе [2].

Для решения данной задачи нами было принято решение о разработке информационной системы акустических расчетов и исследований, которая позволила бы оперативно и качественно их выполнять и позволяла студентам овладевать методами обработки акустических данных и проводить их измерения.

После обзора специализированной и учебной литературы нами определялись задачи, которые будут решаться разрабатываемой системой. Разрабатываемая информационная система будет содержать семь задач, разделенных на три логических группы:

1. Группа: Задачи на вычисление звуковых волн.
1.1. Задача: Вычисление отражения волны с постоянным импедансом.
1.2. Задача: Прохождение звуковой волны.
1.3. Задача: Расчет кривой звукопоглощения;
2. Группа: Задачи вычислений свойств звука.
2.1. Задача: Вычисление уровня звукового давления.
2.2. Задача: Вычисление спектра собственных частот помещения.
2.3. Задача: Вычисление уровня звука;
3. Группа: Остальные задачи.
3.1. Задача: Вычисление времени реверберации помещения.

Теперь, когда основное содержание информационной системы было определено, нами были определены подходящие средства разработки.

В качестве языка программирования был выбран Python. Основными его достоинствами являются:

1. Высокий уровень абстракции — позволяет программисту думать только о вещах, связанных с программой напрямую;
2. Простота написания кода — в комбинации с предыдущим пунктом, позволяет ускорить сроки разработки;
3. Кроссплатформенность — одно из самых сильных преимуществ языка, позволяет запускать программы на абсолютно разных машинах и системах, без изменений исходной программы.

В качестве графической системы была выбрана библиотека PySide6, которая предоставляет программисту широкий спектр возможностей (динамическое изменение интерфейса, анимации и т.п.) [3]. Также безусловным достоинством данной графической среды является кроссплатформенность (поддержана в большинстве популярных операционных систем). Помимо всего это PySide6 обладает «хорошей» (более дружелюбной к разработчикам) лицензией LGPL.

Для визуализации результирующих данных (создание и отображение графиков) будет использована библиотека pyqtgraph, имеющая интеграцию с PySide6.

В качестве среды разработки была выбрана и использована PyCharm (CommunityEdition 2022) от компании JetBrains.

Для реализации указанных задач, было осуществлено проектирование архитектуры информационной системы.

Ввиду того, что разрабатываемая информационная система будет решать ограниченное число задач в сфере акустических вычислений, предлагается разработать модульный «движок», позволяющий структурировать систему на различные модули и как следствие загружать их.

Концептуально предлагается информационную систему разделить на три независимые части (см. рисунок 1):

1. Система модулей/задач;
2. Шаблонное (базовое) приложение, с которым модули будут взаимодействовать;
3. Внешняя система, которая будет включать в себя функции, слишком специфичные для внесения внутрь «движка» (например, подсистема отображения графиков и диаграмм).

Такая архитектура позволит:

— во-первых, правильно структурировать код;
— во-вторых, увеличит модифицируемость, гибкость и расширяемость итогового приложения.

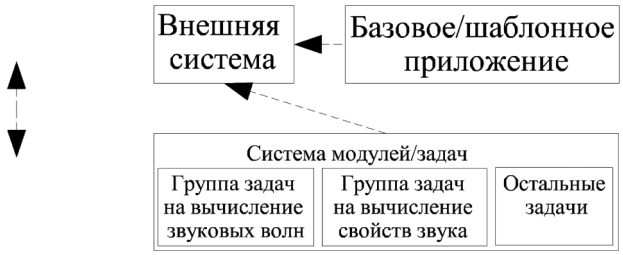

Рис. 1. Архитектура информационной системы

Говоря о системе модулей необходимо отметить, что они будут разделены на интерфейсную (frontend) и вычислительную (backend) составляющие. Также потребуется класс объединяющий интерфейс и вычисления (см. рисунок 2). Данное структура модулей программы обусловлена архитектурой системы модулей MVC (model — view — controller):

1. Model — вычислительная составляющая (backend);
2. View — интерфейсная составляющая (frontend);
3. Controller — объединяющий класс.

Модули будут объединены в группы по общим признакам (например, задачи на вычисление параметров звуковой волны).

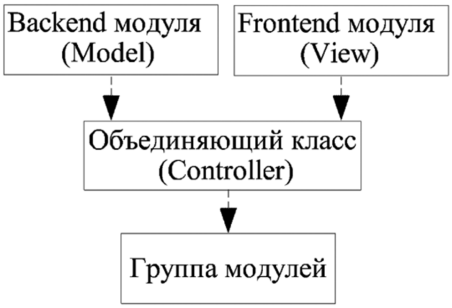

Рис. 2. Архитектура системы модулей

Также данный «движок» будет предоставлять загрузчик модулей.

Помимо вышеперечисленного, система модулей имеет внутреннюю подсистему, реализующую следующие структуры данных: контроллер модулей; Frontend модулей; Backend модулей; группу модулей; базовую сущность; результат вычислений и входные параметры задачи.

Шаблонное (базовое) приложение будет предоставлять «чистое поле» на котором модули будут размещать свои данные в специализированных зонах.

На данный момент в приложении будут присутствовать следующие зоны:

1. Зона списка задач;
2. Зона модуля;
3. Зона результатов модуля;
4. Зона краткого описания задачи.

Также базовое приложение будет реализовывать различный функционал отображения (например, анимации и собственные объекты для отображения результатов).

Как было описано ранее внешняя система будет включать вещи слишком специфические для того, чтобы включить их в ядро «движка», например, систему создания диаграмм, так как:

— во-первых, не каждому приложению могут потребоваться графики;
— во-вторых, они создаются с помощью библиотеки pyqtgraph, которая в свою очередь зависит от библиотеки PySide6, что очень сильно уменьшает применимость разработанного «движка» в других приложениях и существенно повышает размер ядра (с учетом зависимостей) — для обобщенной системы коей и является наш «движок» — это недопустимо.

Также в данную подсистему будут входить различные структуры данных, разработанные для взаимодействия с диаграммами и модулями:

1. Представление диаграммы;
2. Линейная диаграмма;
3. Столбчатая диаграмма;
4. Данные диаграммы;
5. Опции диаграммы;
6. Параметры модуля;
7. Опции модуля.

Также внешняя подсистема реализовывает функционал взаимодействия между ядром «движка» в виде загрузчика модулей/задач и самим приложением с помощью функционала, названного межсистемными взаимодействиями (например, отображение интерфейса модуля). Так как данные действия имеют зависимости, как минимум от графического фреймворка.

На первом этапе реализации было создано шаблонное (базовое) приложение, которое будет предоставлять

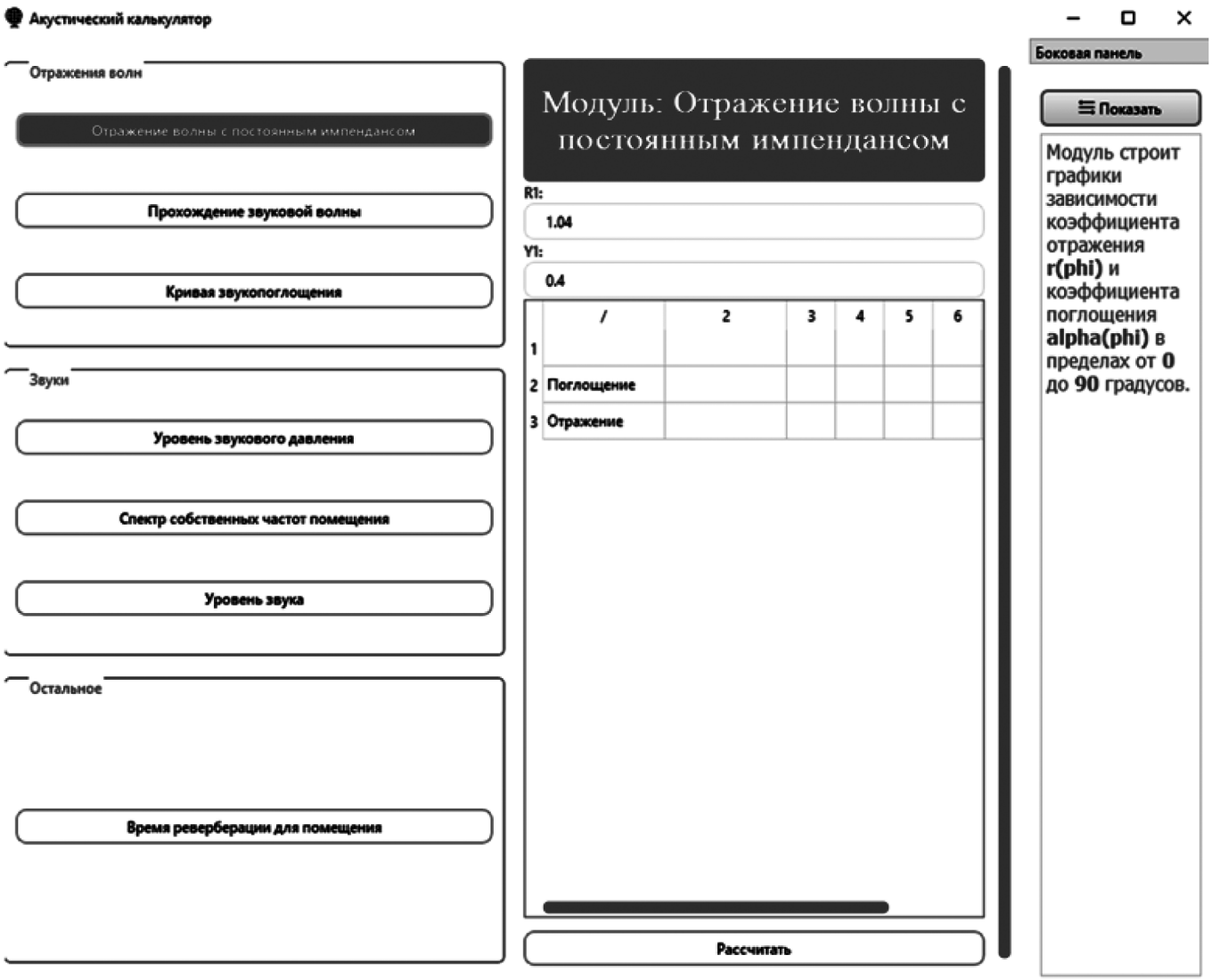

Рис. 3. Специализированные зоны шаблонного приложения

специализированные зоны, которые загружаемые модули будут заполнять и изменять по своему усмотрению (см. рисунок 3).

Рассмотрим каждую из зон более подробно:

1. Зона размещения групп и модулей;
2. Зона интерфейса модуля;
3. Зона описания модуля.

Как видно из названия, в первой зоне отображаются группы и принадлежащие им модули, при нажатии на соответствующую кнопку будет отображен интерфейс запрошенного модуля.

Зона интерфейса модуля является «безразмерной» и имеет полосу прокрутки, что снимает всякие ограничения на отображение интерфейса и результата модуля.

Зона описания модуля заполняется автоматически при отображении соответствующего модуля. Кнопка «Показать» расширяет зону описания и зону модуля для более детального отображения.

Также в базовом приложении реализованы дополнительные элементы интерфейса для отображения результатов:

1. QResultLabel — поле надписи, для отображения результата. При отображении результата проигрывается анимация «подбора» значения (путем последовательного отображения случайных чисел каждые 300 миллисекунд);
2. QTitleLabel — стилизованное поле надписи для отображения названия модуля;
3. QResultTableWidgetItem — поле таблицы предназначенное для отображения результата, имеют аналогичную с QResultLabel анимацию;
4. QInfoBrowser — поле текста, предназначенное для отображения описания модуля;
5. QFloatEdit — поле ввода данных, реализующее ввод чисел с плавающей точкой;
6. QIntEdit — поле ввода данных реализующее ввод целых чисел.

Для реализации системы модулей потребуются соответствующие структуры данных (см. рисунок 4):

1. ProblemResult — структура данных, хранящая именованные результаты вычислений;
2. ProblemArgs — структура данных, хранящая входные данные задачи;
3. ProblemTool — абстрактная структура данных, реализующая итерирование по элементам.

Перечисленные выше структуры данных являются наследниками ProblemTool и как следствие поддерживает итерирование по внутренним элементам.

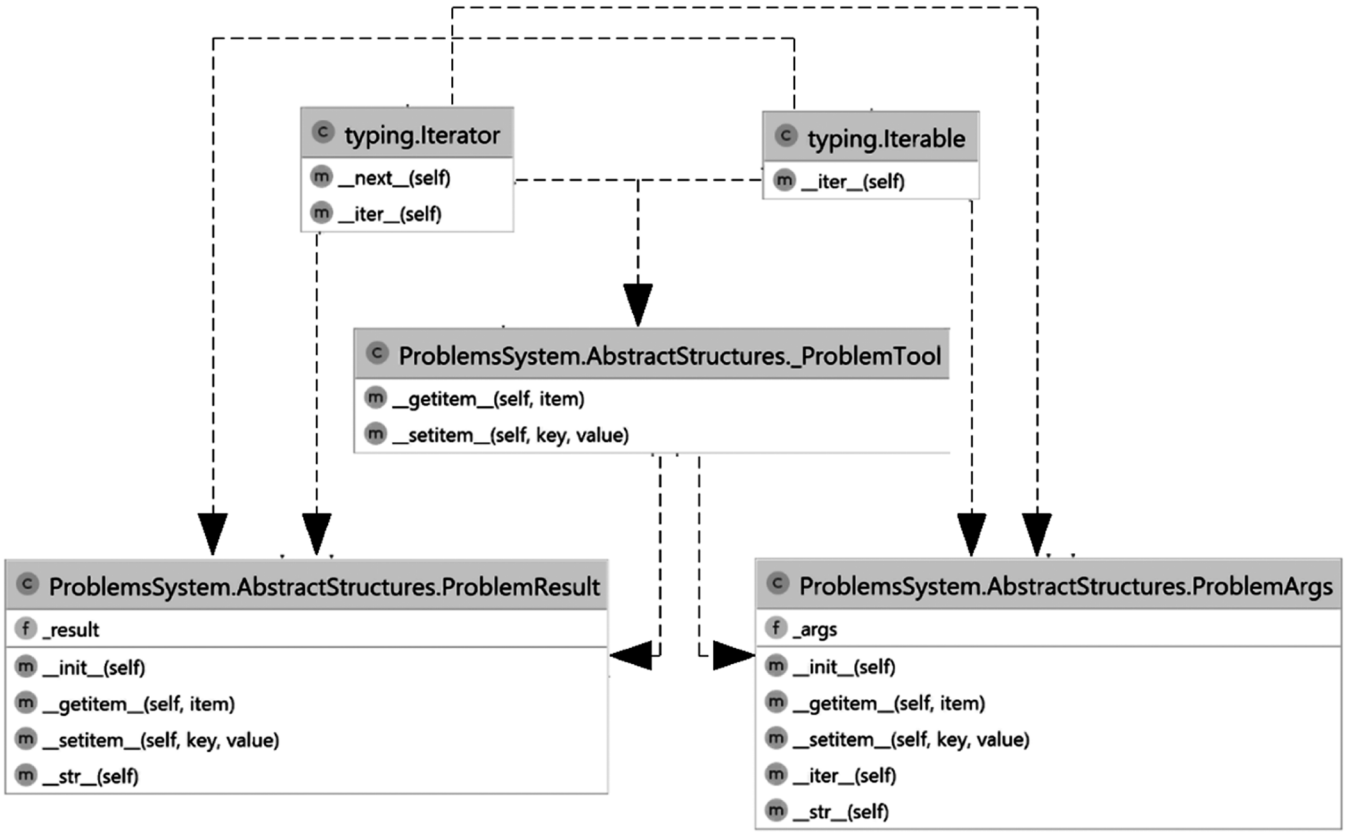

Рис. 4. Структуры данных

Система модулей также содержит соответствующие структуры данных для задачи и групп задач, обе этих структуры данных наследуются от класса сущность и реализуют интерфейс для получения имени и описания соответствующей сущности (см. рисунок 5):

1. Entity — базовый абстрактный класс, реализует методы получения имени и описания;
2. Problem— класс для реализации задачи, содержит следующие методы и свойство:

2.1. Backend — свойство, реализующее вычислительную модель задачи;

2.2. Frontend — свойство, реализующее интерфейс задачи;

2.3. Clear_result — метод, очищающий поле результата;

2.4. Set_result_action — метод, устанавливающий основной действие задачи на соответствующий элемент интерфейса;

2.5. Result_action — метод, выполняющий очистку интерфейса и последующее вычисление и заполнение результатов;

2.6. Solve_problem — метод, выполняющий вычисление результатов;

3. ProblemGroup — класс, реализующий группу задач.

При проектировании было решено разделить вычисления задачи и её интерфейс (frontendи backend), для этого также потребуется реализовать соответствующие структуры данных (см. рисунок 6).

IProblemBackend — класс-интерфейс, реализующий один метод solve, принимающий на вход структуру данных ProblemArgs и возвращающий ProblemResult — метод solve реализует связанные с задачей вычисления.

IProblemFrontend — абстрактный класс, реализующий взаимодействие пользователя и задачи, т.е. пользовательский интерфейс. Класс содержит следующие методы:

1. Init_problem — инициализирует интерфейс задачи;
2. Draw_problem_interface — отображает интерфейс задачи;
3. Draw_result_interface — отображает интерфейс результата;
4. Get_problem_args — формирует структуру ProblemArgs из введённых в интерфейс задачи данных;
5. Fill_result — заполняет интерфейс результата задачи;

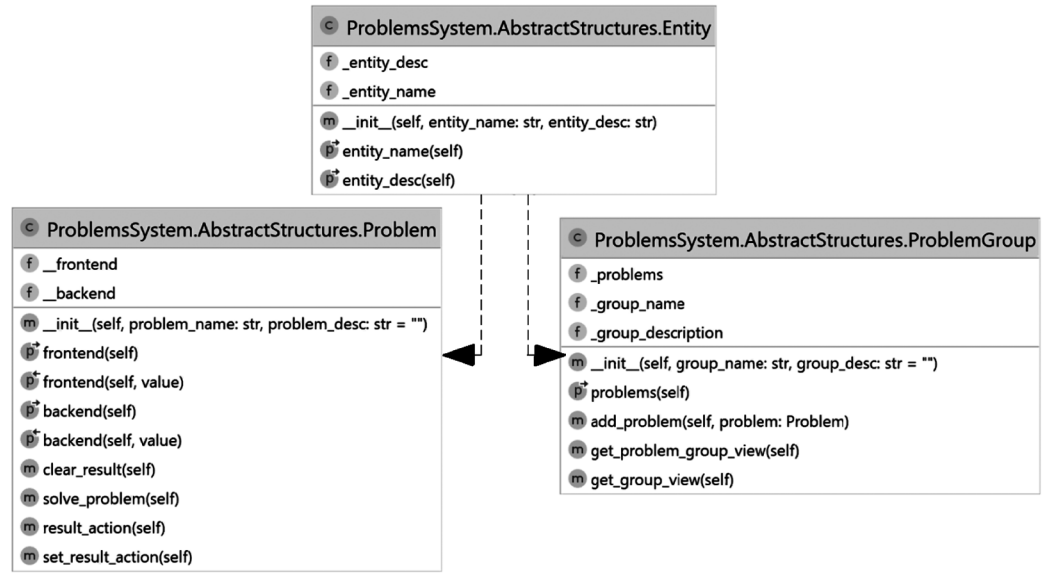

Рис. 5. Диаграмма классов задачи и группы задач

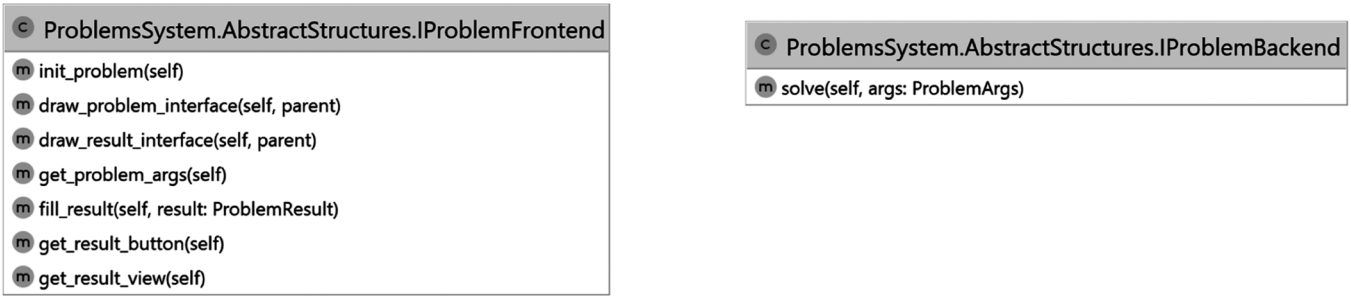

Рис. 6. Диаграмма классов вспомогательной структуры данных задачи

6. Get_result_button — возвращаетэкземпляр кнопки основного действия;
7. Get_result_view — возвращает поле отображения результата, для его последующей очистки.

Внешняя система реализует механизмы и структуры данных, специфичные (имеющие зависимости) для различных систем (см.рисунок 7):

1. ChartData — структура данных для хранения отображения на графике;
2. ProblemOptions — структура данных хранящая различные элементы базового приложения, для доступа к ним внутри модуля задачи;
3. TaskOptions — класс, реализующие различные вспомогательные функции для взаимодействия модулей и базового приложения:

3.1. Clear_module_view — очистка интерфейса модуля;
3.2. Clear_module_chart — очистка диаграммы модуля;
3.3. Clear_all_module_data — очистка всех данных модуля;
3.4. Notify — функция «оповещение» — расширяет зоны модуля и описания.

Помимо вышеперечисленных структур данных внешняя система содержит вспомогательный функционал. Класс ChartView реализует создание диаграмм, на основе переданных в него типов данных.

Классы ChartSeriesи ChartBarSeries реализуют структуры данных, с помощью которых определяется вид диаграммы (линейная или столбчатая) (см. рисунок 8).

Благодаря разработанной ранее архитектуре мы можем перейти к непосредственной реализации задач. Сама реализация модулей (задач) стандартна: они используют описанные ранее структуры данных, а именно классы:

1. IProblemBackend — реализация расчетов задачи;
2. IProblemFrontend — реализация интерфейса задачи;
3. Problem — класс «обёртка» совмещающий в себе предыдущие два.

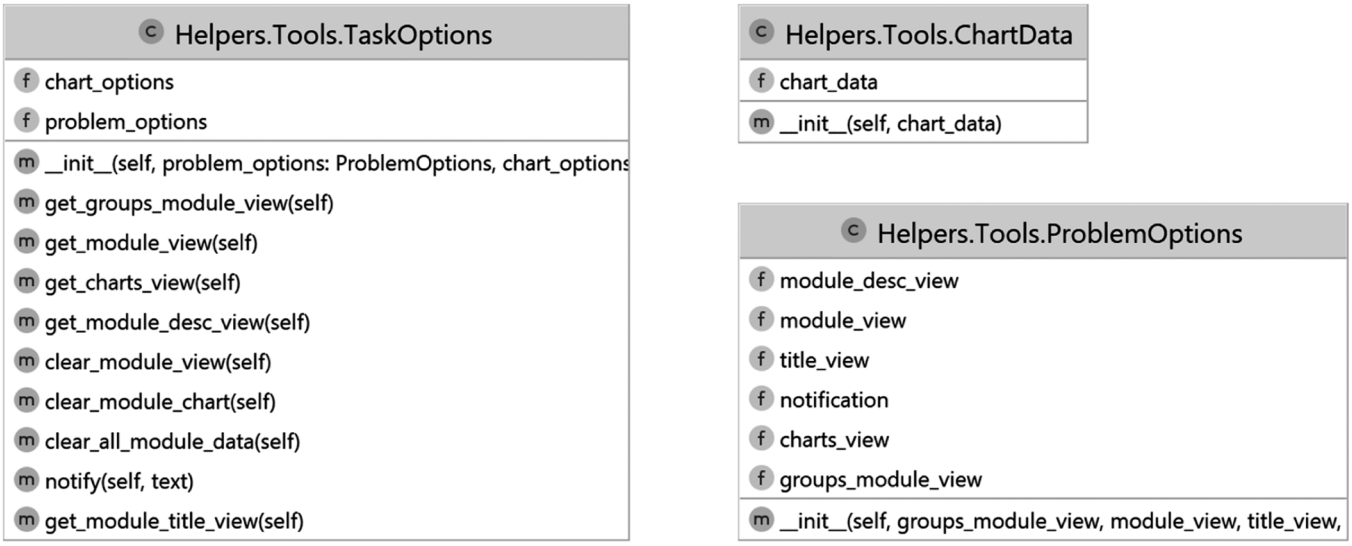

Рис. 7. Диаграмма классов структур данных внешней системы

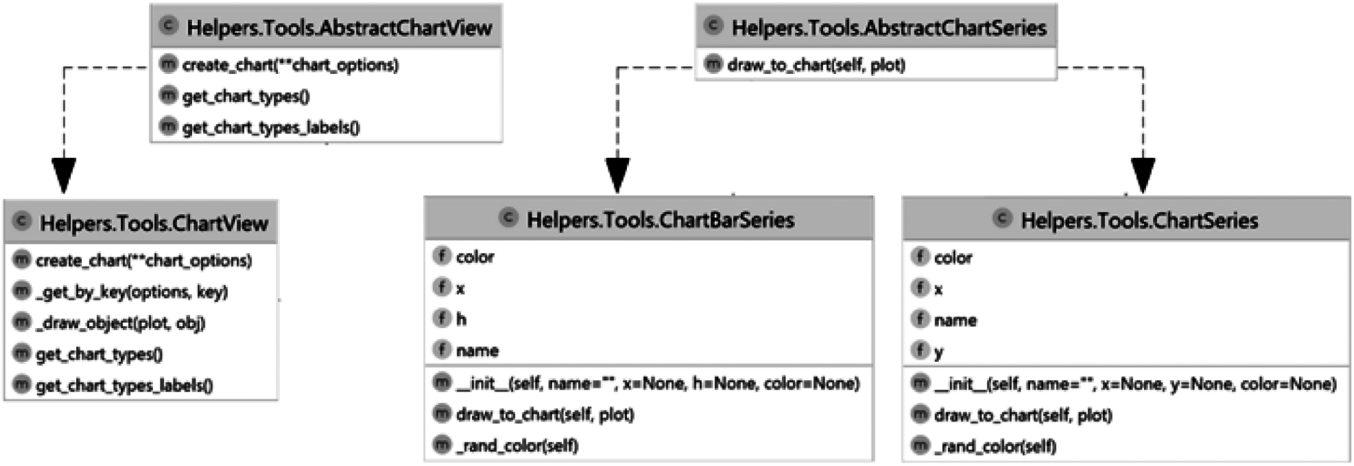

Рис. 8. Диаграмма классов инструментов внешней системы

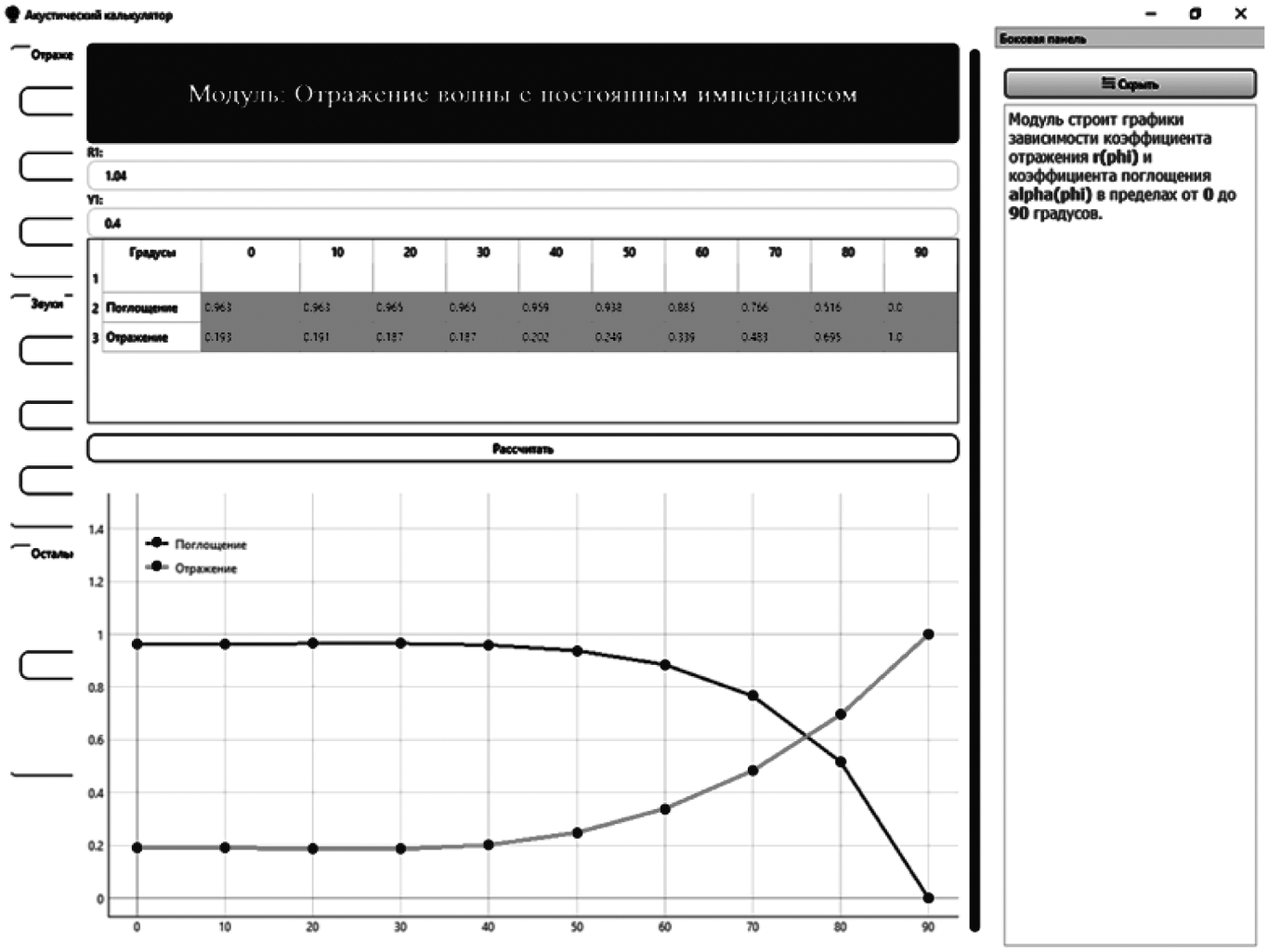

Рис. 9. Вычисление отражения волны с постоянным импедансом

Специфичный для каждой задачи код, заключается в создании уникального для каждой задачи интерфейса и реализации математических расчётов.

На данный момент, как уже было указано ранее, в ИС реализованы заявленные в начале задачи.

После создания всех перечисленных модулей нами было проведено тестирование системы и проверена правильность расчетов (см.рисунок 9).

Все расчеты выполнены правильно и результаты представлены в требуемой форме. Данная система может использоваться, как для акустических расчетов и исследований, так и в учебном процессе.

Плюсом данной системы является возможность отображения результатов в графическом виде, что значительно облегчает усвоение предмета студентами строительных специальностей.

ЛИТЕРАТУРА

1. Вощукова, Е.А. Физические основы строительной акустики: учебноепособие для студентов очного и заочного обучения (направление подготовки бакалавров «Строительство») / Е.А.Вощукова. — Брянск: БГИТА, 2011. — 96 с.
2. Прусс Б.Н., Иванов М.А., Камшило М.П. К вопросу снижения транспортного шума в городской среде//Актуальные направления научных исследований XXI века: теория и практика. — 2015.— Т. 3. № 6 (17). — С. 174–178.
3. Плескачева О.Ю., Прусс Б.Н., Романов В.А., Соболева Г.Н. Разработка информационной системы геодезических расчетов и исследований//Современная наука: актуальные проблемы теории и практики: Серия «Естественные и Технические науки». — № 6.— 2024.— С.110–116.

© Плескачева Ольга Юрьевна (pleskacheva@inbox.ru); Прусс Борис Наумович (prussbor@gmail.com);
Романов Виктор Александрович (vromanov62@mail.ru)
Журнал «Современная наука: актуальные проблемы теории и практики»