



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

**«МИРЭА – Российский технологический университет»
РТУ МИРЭА**

Институт Кибернетики

Кафедра программного обеспечения систем радиоэлектронной аппаратуры
при АО «Концерн «Вега»

Курсовая работа

**Средство интеграции функции определения времени пробега сигнала в
слоисто-однородной среде в задачу сейсмического мониторинга
по дисциплине «УРПО»**

Студент группы КМБО-02-15

Муромцева Л.К.

Преподаватель

Завьялов А.В.

Москва 2018

| | |
|-----------------------------|-----------|
| Общие положения | 3 |
| Назначение программы | 3 |
| Заинтересованные лица | 3 |
| Архитектурные мотивы | 5 |
| Ключевые сценарии качества | 6 |
| Практичность | 6 |
| Модифицируемость | 6 |
| Шаблон архитектуры | 6 |
| Описание архитектуры | 7 |
| Распределение по файлам | 9 |
| Пользовательский интерфейс | 12 |
| Прикладная функциональность | 15 |
| Интеграционная | 16 |
| Ядро системы | 16 |
| Состав компонентов ядра | 18 |
| Стационарные связи | 20 |
| Показатели* | 21 |
| Выводы | 24 |

Общие положения

Назначение программы

Программа предназначена для:

1. Вычисления времени пробега сигнала в слоисто-однородной среде и эпицентрального расстояния события
2. Интеграции функции вычисления времени пробега в общую задачу сейсмического мониторинга (PhsDP)

Общая задача сейсмического мониторинга PhsDP исследует процессы землетрясений и их возникновения. Программное средство позволяет более детально приблизиться к очагу землетрясения и увидеть больше деталей по сравнению с возможностями существующих методов, а также проследить изменение процесса землетрясения во времени.

Заинтересованные лица

| Заинтересованные лица | | Интерес | Точка зрения |
|--|--------------------------|---|----------------|
| 1. Сотрудники ФИЦ Единая геофизическая служба Российской академии наук - первичный потребитель | | | |
| | 1.1 Пользователи системы | 1.1.1 Простота и удобство описания структуры предметной области 1.1.2 Простота и удобство использования системы: наглядная | Интеграционная |

| | | | |
|---|--|--|---|
| | | инструкция, понятная визуализация процесса | |
| | 1.2 Разработчики, использующие систему PhsDP | 1.2.1 Исключение необходимости разработки собственных алгоритмов интеграции | Интеграционная |
| | | 1.2.2 Исключение необходимости разработки деталей интерфейса | Пользовательский интерфейс |
| | | 1.2.3 Гибкость | Ядро, стационарные связи |
| | | 1.2.4. Модифицируемость | Ядро, интеграционная, прикладная функциональность, Стационарные связи |
| | | 1.2.5 Простота реализации | Прикладная функциональность |
| 2. Нефтегазовые и газодобывающие компании - вторичный потребитель | | 2.1. Соответствие программы изначальным требованиям 2.2. Точность полученных результатов | Прикладная функциональность |
| 3. Разработчики системы PhsDP | | 3.1. Гибкость | Ядро, стационарные связи |
| | | 3.2 | Ядро, |

| | | |
|---|---|---|
| | Модифицируемость | интеграционная, прикладная функциональность, стационарные связи |
| 4. Люди, проживающие в сейсмически активных зонах | 4.1 Точное, своевременное получение информации о сейсмически активных зонах | Прикладная функциональность |

Таблица 1. Заинтересованные лица

Архитектурные мотивы

1. В системе необходим удобный в использовании специалистам в предметной области графический интерфейс для управления данными и визуализации данных.
2. Система должна обеспечивать:
 - a. Гибкость
 - b. Модифицируемость
 - c. Перенастраиваемость
 - d. Использование технологии отложенного связывания
 - e. Редактирование графического интерфейса и привязка функциональности к элементу графического интерфейса в программе без повторной компиляции, облегчающие процесс разработки
 - f. Простота внесения изменений
 - g. Простота проверки, контрелепригодность
 - h. Инфраструктура, облегчающая процесс разработки
3. Добавление функциональности во время выполнения не должно замедлять работу системы

Ключевые сценарии качества

Практичность

Время выполнения запрошенных пользователем длительных операций не должно превышать 3 секунд.

Время запуска программы не должно превышать 1 секунды, за исключением времени 1-ого запуска программы.

Среднее время обучения пользователя работе не должно превышать 3-ех дней.

Модифицируемость

Реализация принципа устойчивости абстракций: устойчивый компонент должен быть также и абстрактным, чтобы устойчивость не препятствовала его расширению, т.е. внесение изменений в один модуль, вызванных любым исполнителем, не должно вызывать необходимости внесения изменений в остальные модули.

Необходимо разбить систему на отдельные максимально независимые компоненты, а также функциональные блоки внутри компонент. Изменения в одном функциональном блоке не должны инициировать изменения в другом функциональном блоке.

Шаблон архитектуры

1. Наличие архитектурного мотива по взаимодействию с БД определяет наличие отдельного модуля для задачи взаимодействия с БД.
2. Необходимость динамического добавления функциональности обуславливает наличие модуля, включающий внутреннюю логику программы.

3. Наличие архитектурного мотива п. 2 определяет наличие отдельного модуля, содержащего правила связывания функциональности и графического интерфейса.
4. Наличие графического интерфейса определяет наличие отдельного модуля, в котором объединены классы, реализующие графический интерфейс.

Система интеграции функций сейсмологических вычислений строится по шаблону MVC.

В качестве **View** служат **.xml** файлы, в которых описывается пользовательский интерфейс и привязанная к элементам функциональная составляющая.

Controller-ы - это специальные классы, которые имеют функцию **acceptMsg(char * msg)**. С помощью этой функции происходит общение с GUI и на основе пользовательских действий принимаются решения о том, как взаимодействовать с **Model**.

Model - это классы, содержащие бизнес-логику программы (логику интеграции). Также классы в **Model** взаимодействуют с базой данных.

Описание архитектуры

| Точка зрения | Модель |
|-----------------------------|-----------------------|
| Распределение по файлам | Диаграмма компонентов |
| Пользовательский интерфейс | Диаграмма классов |
| Прикладная функциональность | Текст |

| | |
|--------------------|---------------------------|
| Интеграционная | Текст |
| Ядро | Диаграмма boxes-and-lines |
| Стационарные связи | Диаграмма компонент |

Таблица 2 Описание архитектуры

Распределение по файлам

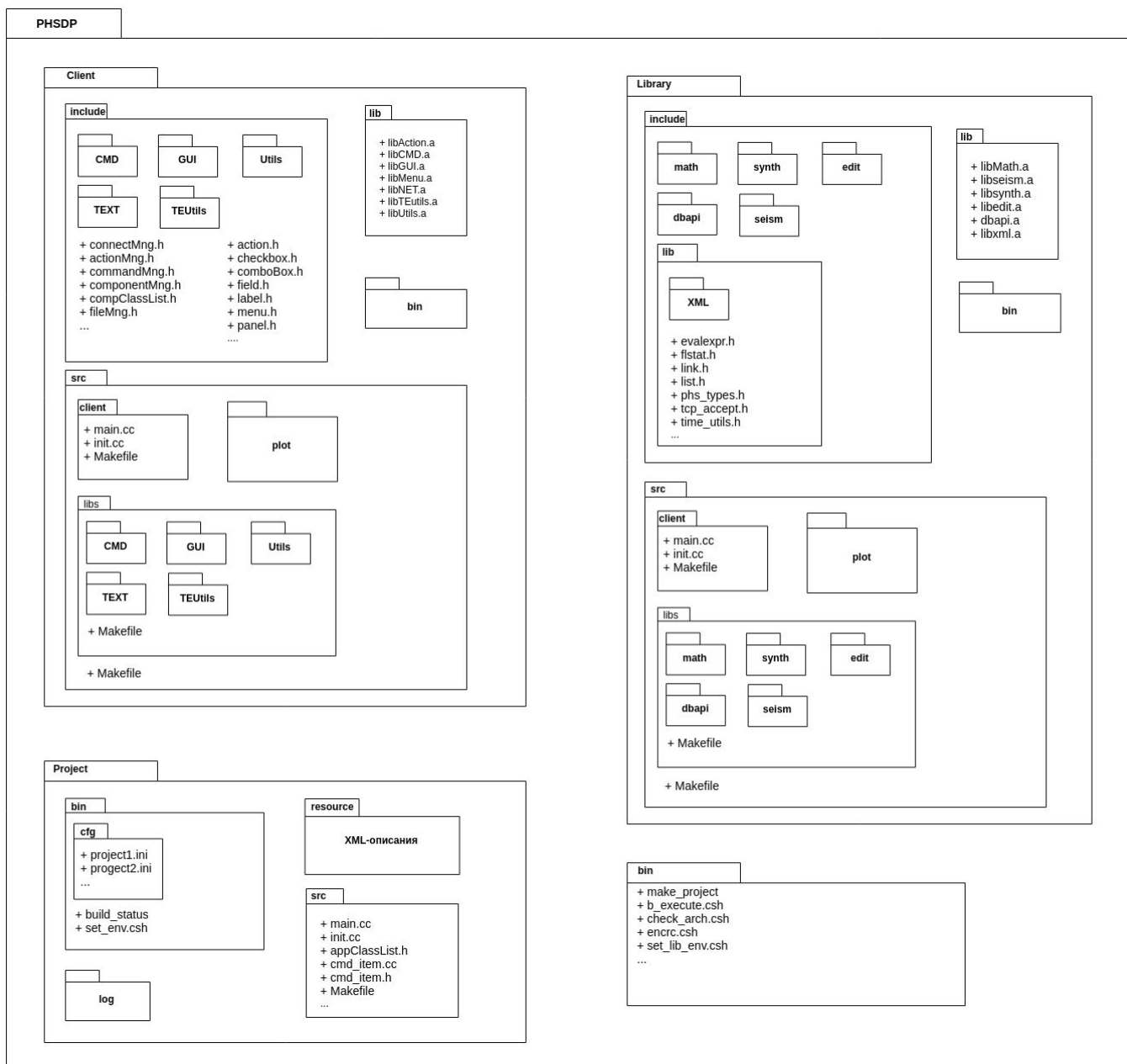


Рисунок 1. Диаграмма пакетов как модель распределения системы по файлам

Функции общего назначения - Library - включает в себя:

1. Математические функции - libmath.a (векторные операции, численные методы)
2. Специальные сейсмологические функции и структуры - libseism.a
3. Функции обработки XML-документов - libxml.a
4. Другие функции, которые содержат в себе пересечения видов вышеизложенных функций

Каталог Include содержит заголовочные файлы для сборки Library.

Каталог Src содержит в себе исходный код для сборки Library.

Функции для обеспечения GUI - Client - включает в себя:

1. Функции обработки сигналов между графическими модулями - libAction.a
2. Функции обработки командной строки, основное предназначение управление внешними задачами и аппаратурой, оборудованной процессорами, - libCMD.a
3. Функции построения основных графических объектов - libGUI.a
4. Функции построения основного графического блока - libMenu.a
5. Функции работы с сетью IP и другими сетевыми соединениями, включая обеспечение асинхронных соединений, соединений RS 232 и др. - libNET.a
6. Функции обработки текста, включает также функции виртуального терминала и полноформатного редактора текста - libTEXT.a
7. Другие функции по работе с выше перечисленными библиотеками и с библиотеками функций на C- libUtils.a

Каталог Include содержит заголовочные файлы для сборки Client.

Каталог Src содержит заголовочные файлы для сборки Client.

Прикладные функции, интегрируемые в систему PHSDP - Project - включает в себя:

1. Интегрируемые функции в систему PHSDP - функции вычисления эпицентрального расстояния, функции вычисления времени сигнала в слоисто-однородной среде.
2. XML-описания графического интерфейса и прикладываемой функциональности.
3. Заголовочный файл, содержащий маппинг элементов, создаваемых в соответствии с XML-описаниями и соответствующих классов.

В файле **.ini** указывается список ресурсных XML-файлов:

- Название XML-элемента в системе
- Название файлов XML-описания элементов
- Настройка параметров главного экрана, таких как размер экрана, формат текста и т.д.

Пользовательский интерфейс

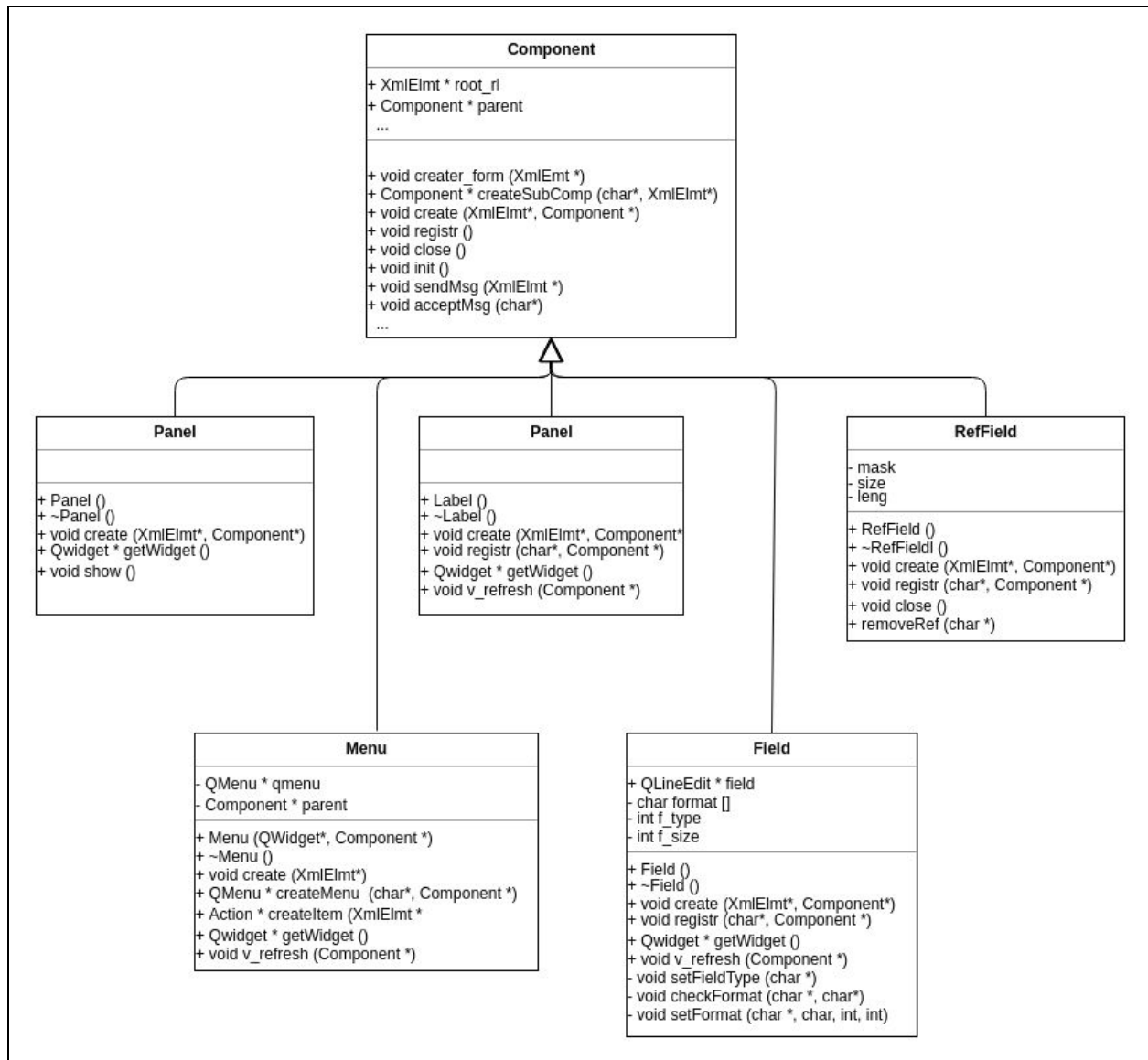


Рисунок 2 Диаграмма основных классов модели пользовательского интерфейса

В компоненте GUI имеются следующие группы классов объектов. Все элементы данного типа являются наследниками абстрактного класса **Component**.

| GUI-элемент | Описание |
|-------------|---|
| Panel | Panel представляет пространство для размещения реализаций классов: panel, label, field, а также других дополнительных структур, таких как кнопки, списки кнопок и др. |
| Label | Поле для отображения информации, модифицируемое только со стороны программы |
| Field | Поле для набора и отображения информации |
| RefField | Поле “внутренней памяти”. Необходимо для хранения адреса объекта или структуры независимо от конкретных функциональных объектов. Особенность данного поля: при изменении значения, все присоединенные элементы получают уведомление о факте изменения. Присоединенный элемент хранит в своей структуре не значение, а ссылку на refField. |
| ExtMemList | Структура полей refField в виде списка |
| Menu | Отдельная структура, которая содержит меню и встраивается в специальное поле menuPanel классов MainWin и MenuPanel |

Таблица 3 Описание основных элементов GUI

| Функция | Описание |
|---|---|
| Component | |
| void create_form (XmlElmt *) | Создание формы XML-элемента |
| Component * createSubComp (char*, XmlElmt*) | Создание вложенного XML-элемента |
| void create (XmlElmt*, Component *) | Создание объекта класса |
| void registr () | Создание иерархической структуры объектов на основе классов, наследуемых от Component |
| void close () | Удаление объекта и чистка памяти |
| void init () | Инициализация объекта |
| void sendMsg (XmlElmt *) | Отправка сообщения объекту определенного XML-элемента |
| void acceptMsg (char*) | Прием сообщений |
| Menu | |
| QMenu * createMenu (char*, Component *) | Создание меню |
| Action * createItem (XmlElmt *) | Создание пункта меню |
| Field | |
| void setFieldType (char *) | Установка типа поля |
| void checkFormat (char *, char*) | Проверка формата текста в поле |
| void setFormat (char *, char, int, int) | Установка формата текста в поле |
| RefField | |
| Void removeRef (char*) | Сигнал об удалении поля RefField. Удаляется только в том случае, если на поле нет ссылок в системе. |

Таблица 4 Описание основной GUI-функциональности

Для выполнения функций отображения на экране используются классы Qt. Классы системы не наследуют виджеты системы (библиотек) Qt, а только управляют ими, для чего в виджеты Qt встраиваются методы, обеспечивающие связь между объектами Qt объектами системы, которые управляют объектами Qt при отображении информации.

Прикладная функциональность

Прикладная функциональность представляет собой набор:

- Заголовочных файлов, в которых описывается:
 - Классы прикладных задач: структура слоистой коры Земли, атрибуты XML-описаний
 - Методы прикладных задач: вычисление времени пробега сигнала в слоисто-однородной среде, расчет эпицентрального расстояния события в Земле
 - Маппинг содержащий маппинг элементов, создаваемых в соответствии с XML-описаниями и соответствующих классов в виде таблицы (appClassList.h)
- Исходных файлов, в которых описывается:
 - Форма и компонент, который регистрируется в системе для возможности дальнейшей коммуникации между компонентами системы
 - Вычислительные функции прикладных задач
 - Функция для приема сообщений, которые система прочитывает в XML-описаниях
- Файлов настройки среды:
 - Определение параметров ОС
 - Установка сторонних библиотечных директорий

- Установка ключей для компиляции сторонних библиотек
- Логи - генерируются автоматически
- Ресурсных файлов - XML-описания используемых элементов с привязкой к функциональности

Интеграционная

Чтобы произвести интеграцию вычислительного алгоритма, необходимо:

1. Создать отдельный проект в директории PHSDP (см. [Распределение по файлам](#)) со структурой изложенной в разделе [Прикладная функциональность](#)
2. Поместить в папку с проектом готовые init.cc и main.cc
3. Разработать функциональность, т.е. дополнить класс Component прикладными функциями.
4. Разработать XML-элемент, описыв его в отдельном XML-файле
5. Разработать XML-файл, включающий в себя структуру программы, интегрируемую в систему PHSDP
6. Дополнить таблицу appClassList.h новыми XML-элементами и прописать в явном виде функции создания компонентов новых классов
7. Скомпилировать проект

Особенности компиляции и запуска проекта находятся в следующем разделе.

Ядро системы

Система сейсмического мониторинга PhsDP построена на базе технологии отложенного связывания (Postpone Connection Technology).

Система состоит из:

- Ядра системы
- Компонентов-утилит, которые определяют функционал системы
- Файлы настройки (в формате XML)

При построении системы необходима компиляция и включение в программу на статическом уровне в обязательном порядке два модуля:

- main.c
- init.c

Остальные компоненты размещаются в библиотеках. Для указания порядка сборки классов из библиотек по умолчанию используется файл appClassList.h, в котором описаны все классы для сборки. В этом модуле в виде таблицы перечислены имена элементов, создаваемых в соответствии с XML описаниями, в связке с адресами функций создающих объекты соответствующих классов из указанных ниже библиотек. Созданием объектов, соответствующих классов, осуществляет один из четырех базовых компонентов системы - ComponentMng.

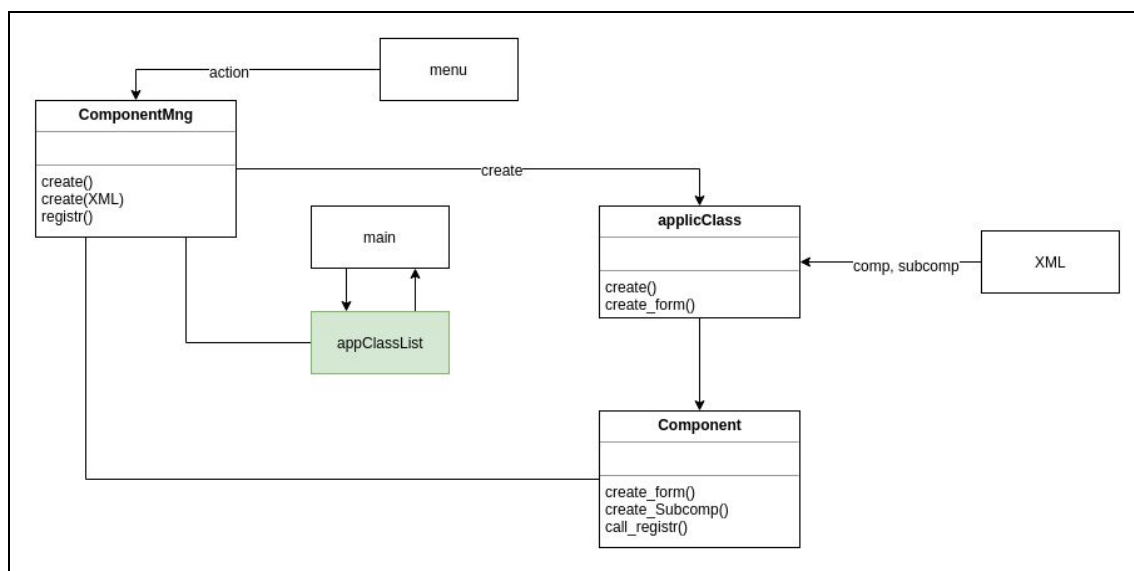


Рисунок 3 Общая схема взаимодействия частей системы: ядра, утилит и файла настройки

Модель (в смысле MVC) описывается в XML-файлах и читается ядром в момент запуска, составляя базу описаний структур потенциальных прикладных классов.

Стационарно собираются программные компоненты ядра:

- ActionMng
- ComponentMng
- CommandMng

Каждая из указанных компонентов запускается в единственном экземпляре. Единственность проверяется на программном уровне.

Объекты классов библиотек, собирающихся не стационарно, запускаются динамически во время запуска программы. Запуск контролируется отдельной частью системы - описанием структур потенциальных прикладных классов и параметрами начальной настройки, описываемыми в виде XML-файлов.

Состав компонентов ядра

ComponentMng - компонент, который отвечает за порождение (вызов конструктора и метода create), а также за вызов деструктора и освобождение ресурсов, выделяемых при создании функциональных блоков.

Основное назначение: управление графическим интерфейсом, построением новых прикладных компонентов по XML-описаниям.

ActionMng - компонент, который отвечает за генерацию сигналов и передачу сообщений между всеми компонентами. Классифицированные сообщения передаются только зарегистрировавшимся в менеджере компонента, а неименованные - передаются по имени, которое указывается в XML файле

настройки, динамически регистрируемом в componentMng'e, регистрируется в таблице с указанием программного (исполняемого) адреса функции acceptMsg().

Основное назначение: обеспечить связь структур меню и кнопок с исполняемыми модулями в синхронном режиме.

CommandMng - компонент, обеспечивающий связь поступающих вне системы данных с системой и обеспечивающая запуск внешних модулей, отслеживание их исполнения и условия их завершения.

Стационарные связи

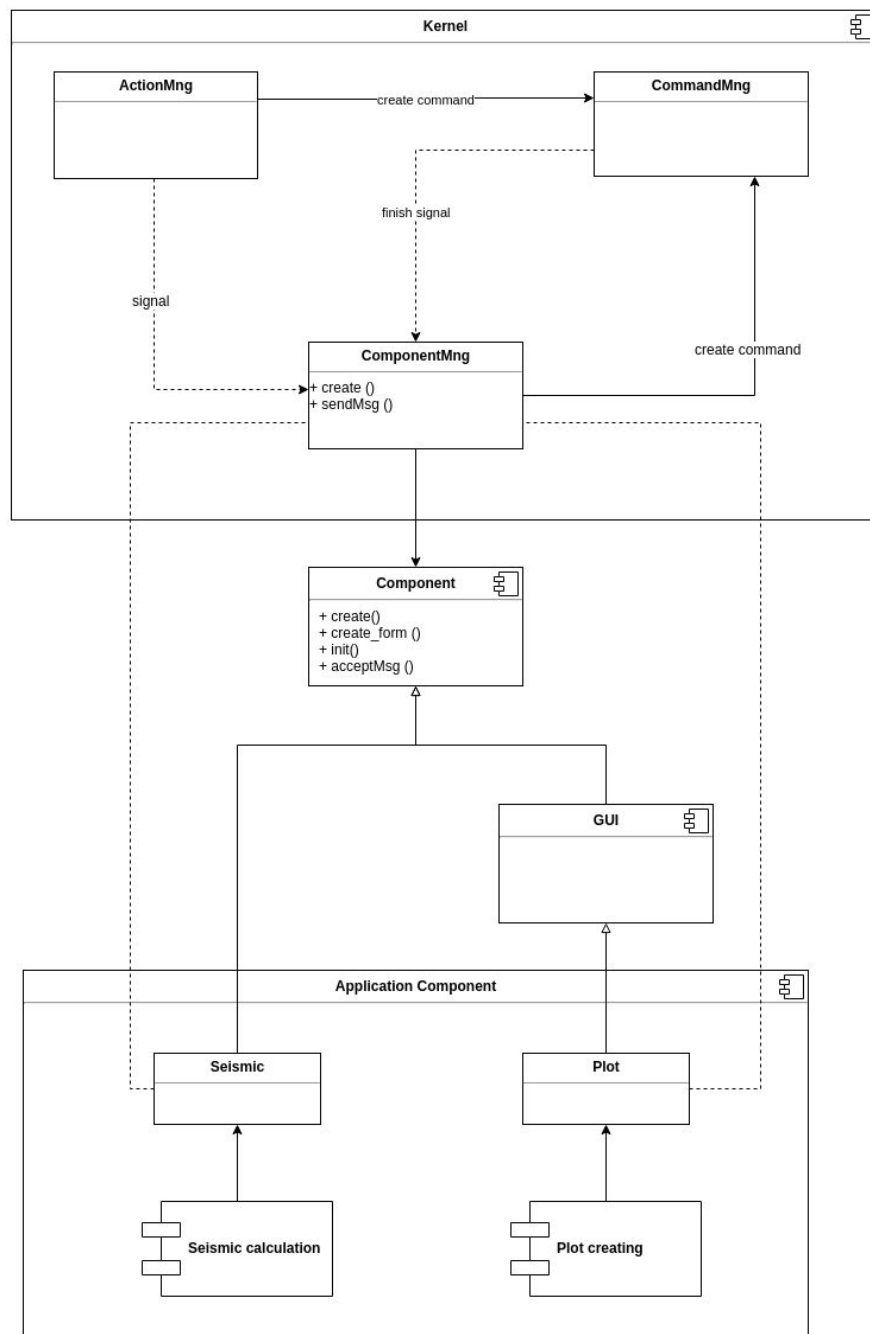


Рисунок 4 Диаграмма компонент системы со стационарными связями

Стационарные связи, в отличие от динамически устанавливаемых связей, устанавливаются в процессе запуска и существуют на протяжении всего времени существования прикладного компонента (Application Component).

В системе существуют следующие компоненты:

1. Kernel - компонент ядра, содержит в себе классы (см. [Состав ядра](#)):
 - a. ActionMng
 - b. ComponentMng
 - c. CommandMng
2. Component - компонент, содержащий в себе функции ядра, используемые при построении цепочки прикладных компонент
3. GUI - компонент, содержащий в себе функции работы с графическим интерфейсом (см. [Пользовательский интерфейс](#))
4. Application Component - прикладной компонент, интегрируемый в общую систему. Содержит в себе классы:
 - a. Seismic - класс, содержащий прикладные параметры (структура земной коры, параметры источника и приемника сейсмического сигнала и т.д.)
 - b. Plot - класс, обеспечивающий отрисовку результатов расчетов Seismic в виде двумерного графика (например, годограф)
 - c. Модуль сейсмических вычислений (расчет времени пробега сигнала в слоисто-однородной среде, расчет эпицентрального расстояния)
 - d. Модуль отрисовки результатов сейсмических расчетовSeismic и Plot взаимодействуют друг с другом через ComponentMng посредством сигналов.

Показатели*

Степень устойчивости (I) - доля входящих зависимостей от их общего числа.

Input - входящие зависимости

Output - исходящие зависимости

$$I = \frac{Input}{Input + Output}$$

Kernel:

$$I_K = \frac{0}{1} = 0$$

Component:

$$I_C = \frac{1}{3} \approx 0.3$$

GUI:

$$I_G = \frac{1}{1} = 1$$

Application Component:

$$I_A = \frac{4}{4} = 1$$

Степень изменчивости (V) - доля исходящих зависимостей от их общего числа

$$V = \frac{Output}{Input + Output}$$

Kernel:

$$V_K = \frac{1}{1} = 0$$

Component:

$$V_C = \frac{2}{3} \approx 0.6$$

GUI:

$$V_G = \frac{0}{1} = 0$$

Application Component:

$$I_A = \frac{0}{4} = 0$$

Степень абстрактности (A) - количество абстрактных классов и интерфейсов от общего числа классов и компонентов

Abstract - количество абстрактных классов

Exact - количество конкретных классов

$$A = \frac{Abstract}{Abstract + Exact}$$

$$A = \frac{1}{7} \approx 0.14$$

Удаленность компонента от главной последовательности

$$D = |A + I - 1|$$

$$D_K = 1$$

$$D_C = 0.3$$

$$D_G = 1$$

$$D_A = 1$$

*Данные показатели не являются корректными с учетом наличия сигналов в системе.

Выводы

В рамках курсового проекта:

1. Проведен системный анализ системы сейсмического мониторинга и технологии отложенного связывания
2. Произведен анализ ядра системы
3. Сформулирован метод интеграции
4. Выявлены ключевые моменты использования системы PHSDP

Система PHSDP на основе технологии отложенного связывания позволяет производить нетрудозатратную, простую и лаконичную интеграцию новых алгоритмов, а также предоставляет возможность сравнивать и оценивать несколько алгоритмов без перекомпиляции проекта для запуска с возможностью вносить изменения в функциональность программы. Внесение изменений в функциональные модули не влияет на уровень модифицируемости системы, т.к. все блоки максимально независимы друг от друга.