Содержание

[Введение 2](#_Toc513498110)

[Моделирование программных систем 3](#_Toc513498111)

[Описание СКО. 5](#_Toc513498112)

[Описание ССО. 7](#_Toc513498113)

[Назначение модуля адаптации. Определение конфигураций системы. 9](#_Toc513498114)

[Описание способа реализации модуля адаптации. 13](#_Toc513498115)

[Вывод 15](#_Toc513498116)

[Список литературы 16](#_Toc513498117)

# Введение

Разработчики современных программных систем постоянно сталкиваются с увеличением уровня их сложности. Это связано с повышением числа требований к программным системам, применением более сложных архитектурных решений и с увеличением объемов программного кода [1].

Задача разработчика заключается в том, чтобы написать правильный код минимального объема. При таком подходе разработка качественно­го программного обеспечения сводится к вопросам выбора архитек­туры, подходящего инструментария и средств управления процес­сом.

Моделирование служит для разработки отдельных аспектов программных систем: контекста выполнения, взаимодействий, структуры и поведения системы. Оно важно для детализации требований к программному обеспечению, а также для проектирования общей архитектуры системы и её отдельных элементов.

Цель данной работы состоит в создании модели модульной программной системы обработки данных с механизмом расчёта на основе конфигурационного подхода.

Работа включает в себя:

* Рассмотрение вопроса необходимости создания модели программной системы;
* описание основ методов адаптации программных систем, моделирования, конструирования, проектирования информационных систем и технологий;
* описание проектируемых систем;
* разработка модуля адаптации на основе конфигурационного моделирования;
* описание и графическое представление модели разрабатываемой системы.

# Моделирование программных систем

Проектирование программной системы начинается с построения модели, описывающей систему с разных точек зрения. Полная модель складывается из отдельных проекций, отражающих разные аспекты системы. Ключом к построению проекций является абстрагирование, когда делается акцент на наиболее существенных деталях, игнорируя при этом менее важные.

Модели позволяют свести высокую сложность программных систем до уровня, понимаемого человеком. Это достигается за счёт иерархического принципа их построения и применения наглядной графической нотации. В процессе построения модели принимаются наиболее ответственные архитектурные решения, поскольку для этого не требуется детальное кодирование. Таким образом, происходит разделение труда: моделирование системы выполняется наиболее квалифицированными разработчиками, способными принять и обосновать кардинальные решения, программную реализацию могут осуществлять программисты среднего уровня, воплощающие принятые и отраженные в модели системные решения в программных кодах [1].

С целью улучшения качества архитектуры программных продуктов в процессе их проектирования был разработан ряд конструкций, применение которых способствует увеличению производительности труда программиста и улучшению структуры программы. Это достигается за счет использования слабосвязанных модульных структур и возможности повторного использования составных модулей [2].

Одним из наиболее популярных методов проектирования адаптивных программных систем является построение так называемой модели пользователя, на основании которой и выстраивается процесс адаптации [3].

Каждая система, каждый интерфейс предназначен для использования, как правило, несколькими категориями пользователей, которые обладают определенными характеристиками. Процесс определения этих характеристик, создание своеобразных «портретов» пользователей, является обязательным этапом проектирования. Этот этап получил название «Определение пользовательских профилей» [4].

Среди параметров модели пользова­теля можно выделить следующие: цель пользователя – параметр, завися­щий от самой природы работы пользователя в системе; уровень подготовки пользователя; имеющийся опыт работы пользователя с данной системой.

В результате работ по определению моделей пользователя разработчики получают описание основных категорий пользователей. Точное их количество зависит от назначения продукта и возможных подходов к определению классов в модели: стереотипный, при котором принадлежность пользователя ограничивается определенным числом классов, и индивидуальный, при котором каждому пользователю соответствует своя модель [5].

В качестве моделируемого объекта возьмём две модульные программные системы с пользовательским интерфейсом, предназначение которых – обработка .xml файлов, хранящих в себе структурированные таблицы данных. Первая система – система серверной обработки файлов с вычислительным механизмом (ССО). Вторая система – система клиентской обработки файлов (СКО). Рассмотрим их более подробно.

# Описание СКО.

СКО включает в себя следующие модули (рис.1):

1) Модуль интерфейса объединения данных;

2) модуль доступа к данным;

3) модуль объединения файлов;

4) модуль проверки данных;

5) модуль хранения данных.

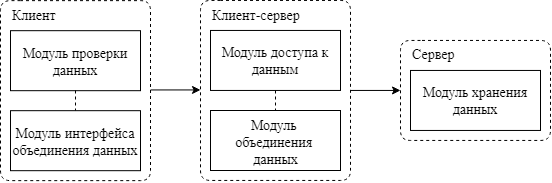


Рис. 1. Структурная схема СКО

Клиентская часть СКО содержит следующие модули:

1) Модуль интерфейса объединения данных, реализующий задачи:

* Предоставление пользователю информации об отправляемом на сервер файле
* Предоставление пользователю возможности задания нового имени серверного файла
* Предоставление пользователю информации о состоянии отправляемого файла
* Предоставление пользователю информации о состоянии процесса удаления данных

2) Модуль проверки данных, который реализует задачи:

* Обеспечение загрузки файла с корректным именем
* Обеспечение загрузки файла с требуемым расширением
* Обеспечение загрузки файла с одной из двух возможных категорий
* Обеспечение наличия загружаемого файла

Клиент-серверная часть системы состоит из описанных ниже модулей:

1) Модуль доступа к данным:

* Предоставление пользователю возможности создавать файлы на сервере
* Предоставление пользователю возможности чтения данных из серверного файла
* Предоставление пользователю возможности добавить данные в серверный файл

2) Модуль объединения файлов:

* Объединение файлов первой категории
* Объединение файлов второй категории

Серверная часть системы содержит модуль хранения данных, реализующий задачи:

* Обеспечение сохранности файлов на удаленном сервере
* Обеспечение механизма обмена данными между сервером и локальным компьютером

Задача системы – предоставление пользователю возможности слияния нескольких файлов со строгой структурой организации данных и отправки результирующего файла на сервер. В ходе работы, выбранные пользователем файлы для загрузки должны пройти все необходимые проверки (корректность и категория содержащихся в них данных, формат файла и др.). В случае неудачной проверки, файл не должен отправляться на сервер. Таким образом, пользователь имеет возможность создания и редактирования рабочих файлов с дальнейшей отправкой их на сервер. На выходе пользователь получает единый файл с объединенными данными для последующего проведения расчётов.

# Описание ССО.

Перечислим список модулей, входящих в состав ССО (рис.1):

1) Модуль интерфейса расчёта данных;

2) модуль доступа к данным;

3) модуль расчёта данных;

4) модуль хранения данных;

5) модуль проверки данных;

6) модуль удаления некорректных данных.

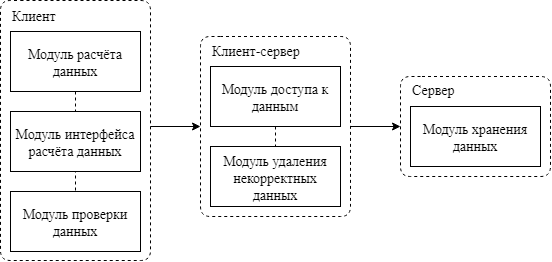


Рис.1. Структурная схема ССО

Клиентская часть системы состоит из трёх модулей:

1) Модуль проверки данных реализующие задачи, перечисленные ниже:

* Обеспечение загрузки файла с корректным именем
* Обеспечение загрузки файла с требуемым расширением
* Обеспечение загрузки файла с одной из двух возможных категорий
* Обеспечение наличия загружаемого файла

2) Модуль интерфейса расчёта данных, который реализует задачи:

* Предоставление пользователю возможности выбора двух опций работы с файлами
* Предоставление пользователю возможности выбора локального файла
* Отображение статуса выполнения расчёта
* Отображение пользователю основных данных для выполнения расчёта

3) Модуль расчёта данных реализует такие задачи как:

* Обеспечение быстрого и точного расчёта данных первой категории
* Обеспечение быстрого и точного расчёта данных второй категории
* Предоставление пользователю результирующих данных в удобном формате

Серверная часть системы содержит в себе модуль хранения данных. Он реализует задачи:

* Обеспечение сохранности файлов на удаленном сервере
* Обеспечение механизма обмена данными между сервером и локальным компьютером

Клиент-серверная часть системы содержит в себе:

1. Модуль доступа к данным, реализующий следующие задачи:

* Предоставление пользователю возможности скачивания серверного файла
* Предоставление пользователю возможности удалить серверный файл
* Предоставление пользователю возможности чтения данных из серверного файла

1. Модуль удаления некорректных данных, предназначенных для реализации перечисленных ниже задач:

* Сохранение в серверном файле только корректных данных
* Удаление некорректного серверного файла

Таким образом, ССО предоставляет пользователю возможность выполнения сложных математических расчётов на основе данных, полученных из серверного файла. Расчёты могут выполняться для двух имеющихся категорий данных, поэтому в системе предусмотрен механизм выбора необходимой цепочки вычислений в зависимости от типа данных из полученного программой файла. В результате пользователь получает файл с рассчитанными данными, доступный для скачивания на клиентский компьютер.

При более подробном рассмотрении можно заметить, что данные системы частично имеют общую функциональность, реализованную в модулях (например, модуль проверки данных, модуль хранения данных и т.д.) Более того исходя из бизнес-логики систем, выходные данные СКО используются в качестве входных для ССО.

С экономической точки зрения, а также со стороны проектирования, наиболее рациональным будет объединить две системы в одну, сделав из каждой отдельной системы подсистему (ПКО, ПСО). Для этого введём модуль адаптации.

# Назначение модуля адаптации. Определение конфигураций системы.

Задача адаптации возникает при наличии в объекте или среде неопределенности, изменяющей его функционирование [6].

Основное назначение модуля адаптации – предоставить пользователю единую систему, имеющую несколько конфигураций, доступных для работы, в зависимости от внешних факторов.

Конфигурация – качественная характеристика структуры, определяющая её пространственную, логическую и временную, а также комплексную организацию сложной целостной системы [7].

Представим фактор управления U в виде пары

U = < Ѱ, F>,

где Ѱ – конфигурация;

F – управляющий фактор.

Для новой разрабатываемой системы, обозначенной как «Комплексная система обработки данных» (КСОД), определим в качестве управляющего фактора наличие различных категорий пользователей с разным набором прав.

Под категорией понимают крупный, обычно строго не очерченный класс в его сравнении с другими такими же классами [8]. В рассматриваемой системе выделим три категории пользователей:

1) Пользователь, обладающий доступом к созданию и изменению серверных файлов;

2) пользователь с доступом к удалению/выгрузке информации с сервера и произведению расчётов;

3) администратор, обладающий полными правами и доступами.

В зависимости от категории пользователя, система предоставляет ему ряд возможных операций, определяемых в той или иной конфигурации системы – Ѱ1, Ѱ2, Ѱ3 (рис.4). Для пользователей первой категории (конфигурация Ѱ1) предоставляется механизм конфигурирования и реконфигурирования рабочих файлов. Для пользователей второй категории (конфигурация Ѱ2) предоставляется механизм конфигурирования расчётных данных. Для пользователей третьей категории (конфигурация Ѱ3) предоставляются механизмы конфигурирования и реконфигурирования рабочих файлов, а также механизм конфигурирования расчётных данных.

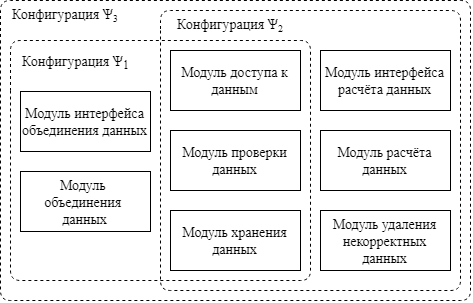


Рис. 4. Схема версий конфигураций программной системы, ориентированных на различные категории пользователей

В ПКО реализован единственный режим конфигурации Ѱ1, в котором пользователь может получать доступ к серверным файлам и объединять их с локальными. Взаимодействие модулей в данном режиме представлено на рис.5.

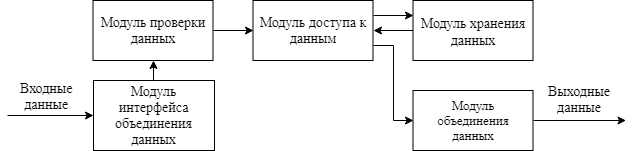


Рис. 5. Структурная схема режима работы конфигурации Ѱ1

В ПСО реализовано два режима конфигурации Ѱ2, состав и взаимодействие модулей в которых представлены на рис.6 и рис.7.



Рис.6. Структурная схема режима работы I конфигурации Ѱ2



Рис.7. Структурная схема режима работы II конфигурации Ѱ2

В режиме I пользователь может скачивать серверные файлы и производить расчёты над данными, полученными из локальных файлов. Режим II позволяет пользователю производить операции удаления некорректных данных из серверных файлов.

Совместная работа двух подсистем позволяет пользователю получить доступ ко всему функционалу, разработанному в рамках КСОД. В данном случае имеет место конфигурация Ѱ3. Она реализует четыре возможных режима работы. Так как режимы I, II, III совпадают с режимами работы конфигураций Ѱ1 иѰ2, рассмотрим более подробно лишь IV режим работы конфигураций Ѱ3 (рис.8.).

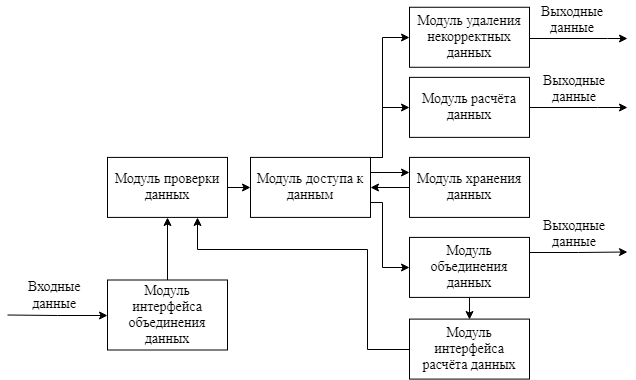


Рис. 8. – Структурная схема режима работы IV конфигурации Ѱ3

В режиме IV конфигурации Ѱ3 пользователь имеет доступ к воспроизведению всех возможных операций: объединять файлы, загружать файлы на сервер и скачивать их с сервера, производить расчёты над данными из локальных файлов и удалять некорректные данные из серверных файлов.

# Описание способа реализации модуля адаптации.

Добавление в систему модуля адаптации позволяет объединить две системы в одну без внесения изменений в их исходный код, тем самым давая возможность обеспечить отсутствие сбоев и ошибок в новой системе, связанных с реорганизацией её структуры.

Существует несколько механизмов адаптации в программных системах [9-11]. Для решения данных задач адаптации будет уместно применить метод конфигурирования. Модуль адаптации представляет собой инструмент для сборки Java проекта – Apache Maven.

Apache Maven – фреймворк для автоматизации сборки проектов на основе описания их структуры в файлах на языке POM, являющемся подмножеством XML [12].

Преимуществами использования Apache Maven являются:

1) Независимость от OS: сборка проекта происходит в любой операционной системе.

2) Управление зависимостями: использование сторонних библиотек, разрешение конфликтов версий.

3) Возможность настройки и выбора разных вариантов конфигураций сборок.

Apache Maven обеспечивает декларативную сборку проекта, т.е. в проектном файле не описывается необходимая последовательность действий по выполнению той или иной цели. Вместо этого, кроме необходимых данных по идентификации данного проекта, описываются необходимые зависимости от библиотек и других модулей проекта, а также указания на использование тех или иных плагинов, влияющих на исполнение целей.

Вся структура проекта будет описываться в файле pom.xml, который должен находиться в корневой папке проекта; операции над другими модулями могут выполняться через него.

Применительно к проектируемой системе, модуль сборки имеет три профиля (handler, calculation и профиль по умолчанию), определяющих конфигурации системы. В зависимости от выбранного профиля, выполняется сборка модулей, указанных в параметрах профилей. При выборе профиля handler сбираются модули, относящиеся к подсистеме клиентской обработки файлов; при выборе профиля calculation – модули подсистемы серверной обработки файлов с вычислительным механизмом. Если профиль не указан напрямую, то по умолчанию выполняется сборка всех модулей системы.

Структурная схема полученной системы КСОД представляет собой совокупность модулей, представленных на рис.9.

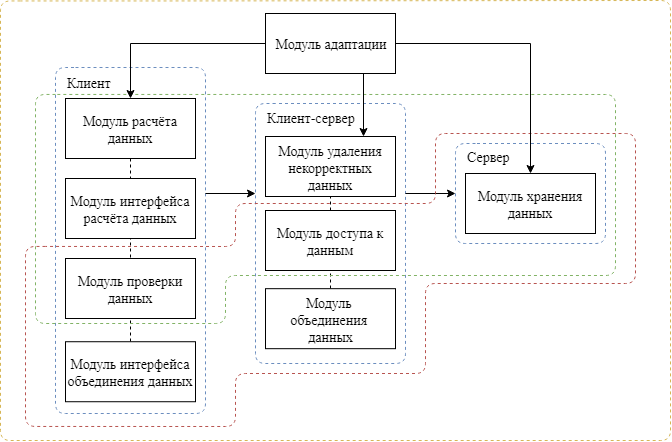


Рис.9. Структурная схема КСОД

# Вывод

В ходе работы был рассмотрен вопрос необходимости создания модели программной системы как один из этапов проектирования, создана и графически представлена модель модульной программной системы КСОД.

В качестве примера был рассмотрен функционал и структура двух систем по обработке и преобразованию данных. По результатам их анализа был сделан вывод о необходимости их объединения в одну систему посредством разработки модуля адаптации на основе конфигурационного моделирования.

Данное решение позволило выделить несколько конфигураций системы, помогающих разграничить доступный функционал системы, в зависимости от прав какой-либо категории пользователей без внесения изменений в программный код.

# Список литературы

1) INFORMICUS – Моделирование программных систем [Электронный источник] http://www.informicus.ru // URL: http://www.informicus.ru/default. aspx?SECTION=6&id=73&subdivisionid=1

2) Ю. И. Величко. Метод внедрения модуля адаптации в пользовательский интерфейс. Проблемы информационных технологий, 2(14):15-19, 2014.3) A. Kobsa. Pseudonymous yet Personalized Interaction with Websites that Utilize Networkwide User Modeling Services. 2003 HCIC Winter Workshop, Winter Park, CO, 2003.

4) Studfiles – Лекция №5 [Электронный источник] https://studfiles.net // URL: https://studfiles.net/preview/5065648/page:3

5) C. C. Валеев, И. М. Исмагилова. Построение адаптивных интерфейсов в сложных распределенных технических системах с применением статистических методов. Математическое моделирование и информационные технологии: Всерос. науч. конф., Екатеринбург, УрГУПС, 3:260-268, 2016.

6) Л. А. Растригин. «Адаптация сложных систем» // Рига: Зинатне, 1981. - 375с.

7) В.В. Нечаев. Конфигурационное моделирование: часть I. Теоретические аспекты: Учебное пособие / Государственное образовательное учреждение высшего профессионального образования «Московский государственный институт радиотехники, электроники и автоматики (технический университет)». – М.: 2007. – 92 с.

8) Ожегов С.И., Шведова Н.Ю. Толковый словарь русского языка: 80 000 слов и фразеологических выражений. – 4-е изд., М.: Высшая школа, 1993. – 944 с.

9) А.М. Бершадский, А.С. Бождай, Ю.И. Евсеева. Механизмы адаптации в современной виртуальной образовательной среде//Интернет и современное общество: сборник тезисов докладов [Электронный ресурс] / Труды XX Международной объединенной научной конференции «Интернет и современное общество» (IMS- 2017), Санкт-Петербург, 21–23 июня 2017 г. — Электрон, дан. — СПб: Университет ИТМО, 2017. — 84 с.

10) Keromytis A.D. Characterizing Software Self-healing Systems // Computer Network Security. Berlin: Springer, 2007. P. 22-33.

11) Michiels S., Desmet L., Joosen W., Verbaeten P. The DiPS+ Software Architecture for Self-healing Protocol Stacks. Proc. 4th Working IEEE/IFIP Conf. on Software Architecture (WICSA'04). 2004, pp. 233-242.

12) Wikipedia – Apache Maven [Электронный ресурс] https://ru.wikipedia.org // URL: https://ru.wikipedia.org/wiki/Apache\_Maven