УДК 004.9

АНАЛИЗ ПРИМЕНИМОСТИ И ПОТЕНЦИАЛЬНОЙ ПОЛЕЗНОСТИ ИСПОЛЬЗОВАНИЯ ШАБЛОНА ПРОЕКТИРОВАНИЯ МОДЕЛЬ-ПРЕДСТАВЛЕНИЕ-КОНТРОЛЛЕР-СЕРВИС ПРИ РАЗРАБОТКЕ АВТОМАТИЗИРОВАННЫХ СИСТЕМ СБОРА И ОБРАБОТКИ ЦИФРОВЫХ ОТЧЁТНЫХ ДОКУМЕНТОВ ДЛЯ ПРОМЫШЛЕННЫХ ПРЕДПРИЯТИЙ

[1]Лясковский В.Л., [2]Федулов А.А.

[1]46-й ЦНИИ Министерства обороны РФ, Москва, Чукотский пр., 10, e-mail: Dop_big@mail.ru

[2]Федеральное государственное унитарное предприятие «Всероссийский научно-исследовательский институт «Центр» (ФГУП «ВНИИ «Центр»), 123242, г. Москва, ул. Садовая — Кудринская, д. 11, строение 1, e-mail: afed2000@yandex.ru

В статье кратко представлен обзор основных архитектурных образцов проектирования – Модель-Представление-Контроллер, Модель-Представление-Представитель, Модель-Представление-Представление-Модель. Более подробно рассмотрен шаблон проектирования Модель-Представление-Контроллер-Сервис. Приведен пример использования архитектурного образца Модель-Представление-Контроллер-Сервис в контексте разработки серверной части автоматизированной системы сбора и обработки цифровых отчетных документов для промышленных предприятий. Выделены основные проблемы разработки автоматизированных систем, в которых не используются шаблоны проектирования. Проведен анализ применимости и потенциальной полезности использования образца Модель-Представление-Контроллер-Сервис в разрезе предметной области и выделены преимущества использования, а также положительное влияние применения данного шаблона проектирования. Таким образом, в работе отражен результат влияния использования архитектурного образца Модель-Представление-Контроллер-Сервис на примере разработки серверной части автоматизированной системы сбора и обработки цифровых отчетных документов для промышленных предприятий.

Ключевые слова: архитектурные шаблоны проектирования, Модель-Представление-Контроллер-Сервис, архитектура автоматизированной системы, разработка автоматизированной системы.

ANALYSIS OF THE APPLICABILITY AND POTENTIAL USEFULNESS OF USING THE MODEL-VIEW-CONTROLLER-SERVICE PATTERN IN THE DEVELOPMENT OF AUTOMATED SYSTEMS FOR COLLECTING AND PROCESSING DIGITAL REPORTING DOCUMENTS FOR INDUSTRIAL ENTERPRISES

[1]Lyaskovsky V.L., [2]Fedulov A.A.

[1]46th Central Research Institute of the Ministry of Defense of the Russian Federation, Moscow, Chukotsky Ave., 10, e-mail: Dop_big@mail.ru

[2]Federal State Unitary Enterprise "All-Russian Research Institute "Center" (FSUE "VNII "Center"), 123242, Moscow, st. Sadovaya - Kudrinskaya, 11, building 1, e-mail: afed2000@yandex.ru

The article briefly provides an overview of the main architectural design patterns - Model-View-Controller, Model-View-Presenter, Model-View-ViewModel. The Model-View-Controller-Service design pattern is discussed in more detail. An example of using the Model-View-Controller-Service architectural pattern is given in the context of developing the server part of an automated system for collecting and processing digital reporting documents for industrial enterprises. The main problems of developing automated systems that do not use design patterns are highlighted. An analysis of the applicability and potential usefulness of using the Model-View-Controller-Service pattern in the context of the subject area was carried out and the advantages of use, as well as the positive impact of using this design pattern, were highlighted. Thus, the work reflects the result of the influence of using the Model-View-Controller-Service architectural pattern using the example of developing the

server part of an automated system for collecting and processing digital reporting documents for industrial enterprises.

Keywords: architectural design patterns, Model-View-Controller-Service, automated system architecture, automated system development.

Введение

Процесс проектирования, разработки и поддержки автоматизированных систем постоянно подвергается значительным изменениям в результате развития новых технологий в области IT. Появление новых подходов, тенденций и парадигм программирования создаёт проблему применения накопленного опыта и навыков разработчиками на практике с течением времени.

Решением является использование готовых архитектурных образцов, которые в свою очередь в меньшей степени подвержены изменениям в результате развития новых технологий. Это происходит за счёт признания сообществом разработчиков вышеописанной проблемы и желанием замедлить её ещё большее развитие.

Архитектурные образцы или, другими словами, шаблоны проектирования – это конструкции, которые позволяют решать типичные задачи, возникающие на разных этапах создания программного обеспечения. Они представляют собой архитектурные подходы, которые направлены не только на решение проблемы вариативности при проектировании и разработке автоматизированных систем, но и на структурирование кода и архитектуры в целом.

Архитектурный шаблон проектирования Модель-Представление-Контроллер-Сервис является расширенным вариантом шаблона Модель-Представление-Контроллер и соответственно обладает большим количеством преимуществ и вариантов реализаций, а также более сложной архитектурной концепцией за счёт добавления нового уровня – уровня сервисов.

Модель-Представление-Контроллер-Сервис применяется в самых разных типах автоматизированных систем, однако, как правило, ориентирован на автоматизированные системы с продвинутой бизнес-логикой.

Обзор основных шаблонов проектирования

Существует несколько похожих шаблонов проектирования, которые предусматривают компонентное разделение:

- Модель-Представление-Контроллер;
- Модель-Представление-Представитель;
- Модель-Представление-Представление-Модель [1, 2].

Каждый из этих шаблонов имеет свою собственную архитектурную концепцию [1, 2].

Отличие архитектурной концепции заключается в непосредственном различии между составными компонентами, а также в различии способов организации взаимодействия между ними [1, 2].

Модель-Представление-Контроллер – шаблон, разделяющий приложение на три уровня:

- модель;
- представление;
- контроллер [1, 2].

Существует два вида архитектурного образца Модель-Представление-Контроллер – Smalltalk и Cocoa [3, 4].

Модель-Представление-Контроллер Smalltalk является традиционной реализацией данного образца, однако в настоящее время большую популярность приобрел шаблон проектирования

Модель-Представление-Контроллер Cocoa, который, как правило, и имеют ввиду при упоминании шаблона

Модель-Представление-Контроллер [3, 4, 5]. Далее речь пойдёт непосредственно о современной интерпретации данного образца проектирования – Модель-Представление-Контроллер Cocoa.

На рисунке 1 приведена схема взаимодействия вышеупомянутых компонентов архитектурного шаблона Модель-Представление-Контроллер Cocoa.

Рис. 1 – Схема взаимодействия компонентов архитектурного шаблона Модель-Представление-Контроллер Cocoa

В данном шаблоне модель непосредственно отвечает за управление данными, т.е., к примеру, она занимается управлением базой данных или какими-либо другими функциями, связанными с базой данных [2, 4, 6, 7]. Представление отвечает за взаимодействие с пользователем, т.е., например, это может быть, как страница, так и данные, которые отправляются пользователю [2, 4, 6, 7]. Контроллер является центральным компонентом шаблона [3]. Он отвечает за логику приложения, манипулирует моделью, получает и обрабатывает данные от пользователя, а также манипулирует представлением [2, 4, 6, 7]. На рисунке 2 схематично изображен архитектурный шаблон Модель-Представление-Контроллер Сосоа с учетом описания уровней.

Рис. 2 – Архитектурный шаблон Модель-Представление-Контроллер Сосоа с описанием уровней

Шаблон проектирования Модель-Представление-Представитель – это архитектурный образец, который является производным от шаблона Модель-Представление-Контроллер и также разделяет приложение на три уровня:

- модель;
- представление;
- представитель [7].

Существует несколько видов шаблона Модель-Представление-Представитель: Модель-Представление-Представитель с контроллером-наблюдателем и Модель-Представление-Представитель с пассивным представлением [7]. Основной идеей архитектурного образца Модель-Представление-Представитель было отделение пользовательского интерфейса от логики, что и было реализовано в шаблоне Модель-Представление-Представитель с пассивным представлением, однако в образце проектирования Модель-Представление-Представитель с контроллером-наблюдателем всё же допускалось представлению знать о модели [7]. Широкого развития данные шаблоны проектирования не получили, тем не менее Модель-Представление-Представитель с пассивным представлением является почти идентичным шаблону Модель-Представление-Контроллер (Модель-Представление-Контроллер Сосоа).

Шаблон проектирования Модель-Представление-Представление-Модель – архитектурный образец, состоящий из трёх уровней:

- модель;
- представление;
- модель представления [7].

Модель-Представление-Представление-Модель является производным шаблоном от Модель представления и также является практически идентичным шаблону Модель-Представление-Контроллер (Модель-Представление-Контроллер Сосоа) [7].

Архитектурные образцы проектирования Модель-Представление-Контроллер, Модель-Представление-Представитель и Модель-Представление-Представление-Модель имеют долгую историю развития, разные вариации и разработаны разными компаниями. Тем не менее их всех объединяет то, что в итоге одна из ветвей развития этих шаблонов проектирования приходит к реализации шаблона проектирования похожего на Модель-Представление-Контроллер (Модель-Представление-Контроллер Сосоа).

Шаблон проектирования Модель-Представление-Контроллер-Сервис

Шаблон проектирования Модель-Представление-Контроллер-Сервис был создан на основе архитектурного образца Модель-Представление-Контроллер, который за счёт своей архитектурной концепции завоевал популярность у разработчиков [6]. В связи с широким распространением шаблона Модель-Представление-Контроллер появилась потребность в его расширении. При использовании архитектурного образца Модель-Представление-Контроллер часто возникала необходимость упрощения реализации контроллеров путем инкапсуляции бизнес-логики. Это позволяло не только наделить контроллеры исключительно возможностью перенаправлять и контролировать выполнение определенных функций, но и избежать дублирования кода благодаря переиспользованию инкапсулированной бизнес-логики [6]. Таким образом появился новый вариант реализации шаблона Модель-Представление-Контроллер – шаблон Модель-Представление-Контроллер-Сервис,

который включал в себя новый уровень – сервисы, содержащий в себе бизнес-логику [6, 8]. На рисунке 3 схематично показан архитектурный шаблон Модель-Представление-Контроллер-Сервис.

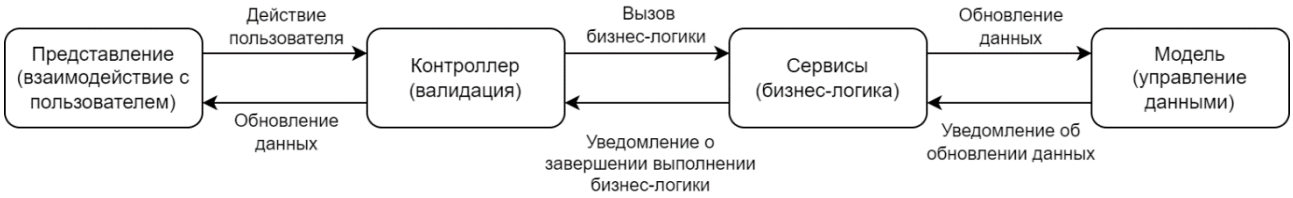

Рис. 3 – Архитектурный шаблон Модель-Представление-Контроллер-Сервис

Стоит сказать и о примерах практического применения данного шаблона проектирования. Один из вариантов архитектурного образца Модель-Представление-Контроллер-Сервис был предложен в статье «Proposal of an architecture for sensor networks monitoring in Open Access Metropolitan Area Networks», где авторы выделяли в шаблоне проектирования Модель-Представление-Контроллер отдельный уровень бизнес-логики, в котором инкапсулировали специфичное для приложения содержимое, а также всю логику обработки данных [9].

В статье «Cross-domain neurobiology data integration and exploration» архитектурный образец Модель-Представление-Контроллер-Сервис лёг в основу системы PubAnatomy и позволил создать высокоинтерактивный пользовательский интерфейс, совместимый практически со всеми браузерами [10]. В настоящее время данный шаблон проектирования используется в компании Microsoft для разработки приложений ASP.NET на языке C# [11].

Архитектурный образец Модель-Представление-Контроллер-Сервис в большей степени подходит для приложений, использующих продвинутую бизнес-логику, так как в данном шаблоне проектирования уровень бизнес-логики четко отделен от остальных уровней [6].

Пример использования шаблона проектирования Модель-Представление-Контроллер-Сервис в разрезе предметной области

Шаблон проектирования Модель-Представление-Контроллер-Сервис в настоящее время активно применяется в различных областях разработки в том числе и в области разработки автоматизированных систем. Примером его применения может стать использование данного архитектурного образца при создании серверной части автоматизированной системы сбора и обработки цифровых отчетных документов для промышленных предприятий.

Разработка собственного шаблона проектирования не имеет смысла, т. к. это усложняет процесс проектирования из-за внедрения новых подходов и абстракций.

Использование уже существующих архитектурных образцов помогает стандартизировать процесс разработки и позволяет не тратить дополнительные временные ресурсы на обучение участников проекта.

Также разработка шаблона проектирования будет требовать дополнительные финансовые вложения. Далее будет представлен пример применения архитектурного образца Модель-Представление-Контроллер-Сервис при разработке автоматизированной системы сбора и обработки цифровых отчетных документов для промышленных предприятий.

Серверная часть автоматизированной системы будет содержать следующие компоненты:

- модели;
- маршруты;
- контроллеры;
- сервисы;
- промежуточное программное обеспечение.

Модели – это часть концепции шаблона проектирования Модель-Представление-Контроллер-Сервис, маршруты являются вариантом реализации представлений из концепции того же архитектурного образца Модель-Представление-Контроллер-Сервис.,

Контроллеры и сервисы – также часть концепции шаблона проектирования Модель-Представление-Контроллер-Сервис, промежуточное программное обеспечения – дополнительный уровень, который инкапсулирует бизнес-логику обработки запросов пользователя, делая контроллер более логически независимым и ориентированным на исключительно валидацию входящих запросов.

На рисунке 4 можно увидеть схематичное изображение применения шаблона проектирования Модель-Представление-Контроллер-Сервис в разрезе предметной области.

Для более глубокой архитектурной декомпозиции необходимо выделить модели, на которых будет строиться автоматизированная система сбора и обработки цифровых отчетных документов для промышленных предприятий.

Будут выделены три основные модели, на которые будет опираться серверная часть разрабатываемой автоматизированной системы:

- отделы;
- задания;
- описания.

Отделы – отделы предприятия, к которым будут привязываться задания. Задания – задания, которые будут содержать в себе базовую информацию о поставленных задачах. Описания – описание заданий, включающее в себя подробную информацию по задаче в виде пары ключ-значение.

Каждая модель должна иметь свой маршрут, контроллер и сервис. На рисунке 5 изображена декомпозиция применения шаблона проектирования Модель-Представление-Контроллер-Сервис в разрезе предметной области.

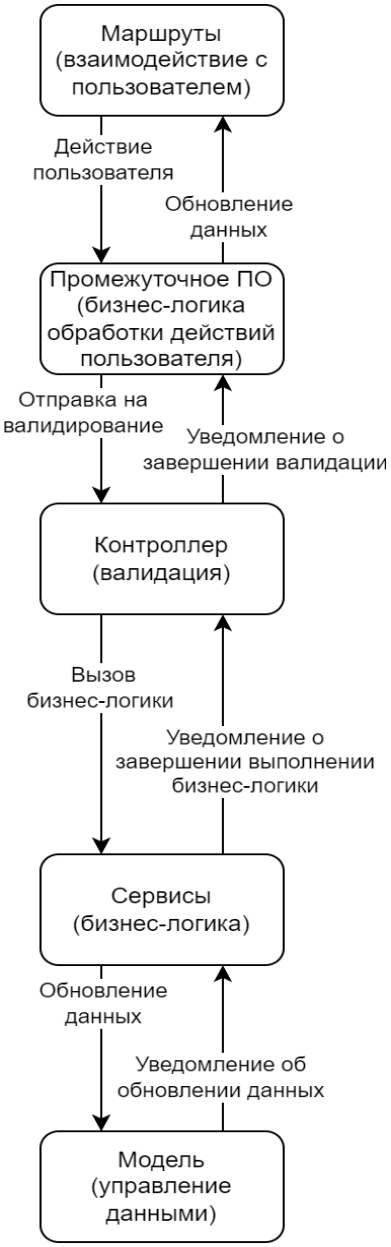

Рис. 4 – Применение шаблона проектирования Модель-Представление-Контроллер-Сервис в разрезе предметной области

Таким образом благодаря применению шаблона проектирования Модель-Представление-Контроллер-Сервис получилось спроектировать архитектуру серверной части автоматизированной системы сбора и обработки цифровых отчетных документов для промышленных предприятий, которая легла в основу создания полноценного клиент-серверного приложения.

На рисунках 6-8 представлено решение, разработанное авторами с использованием архитектурного образца Модель-Представление-Контроллер-Сервис.

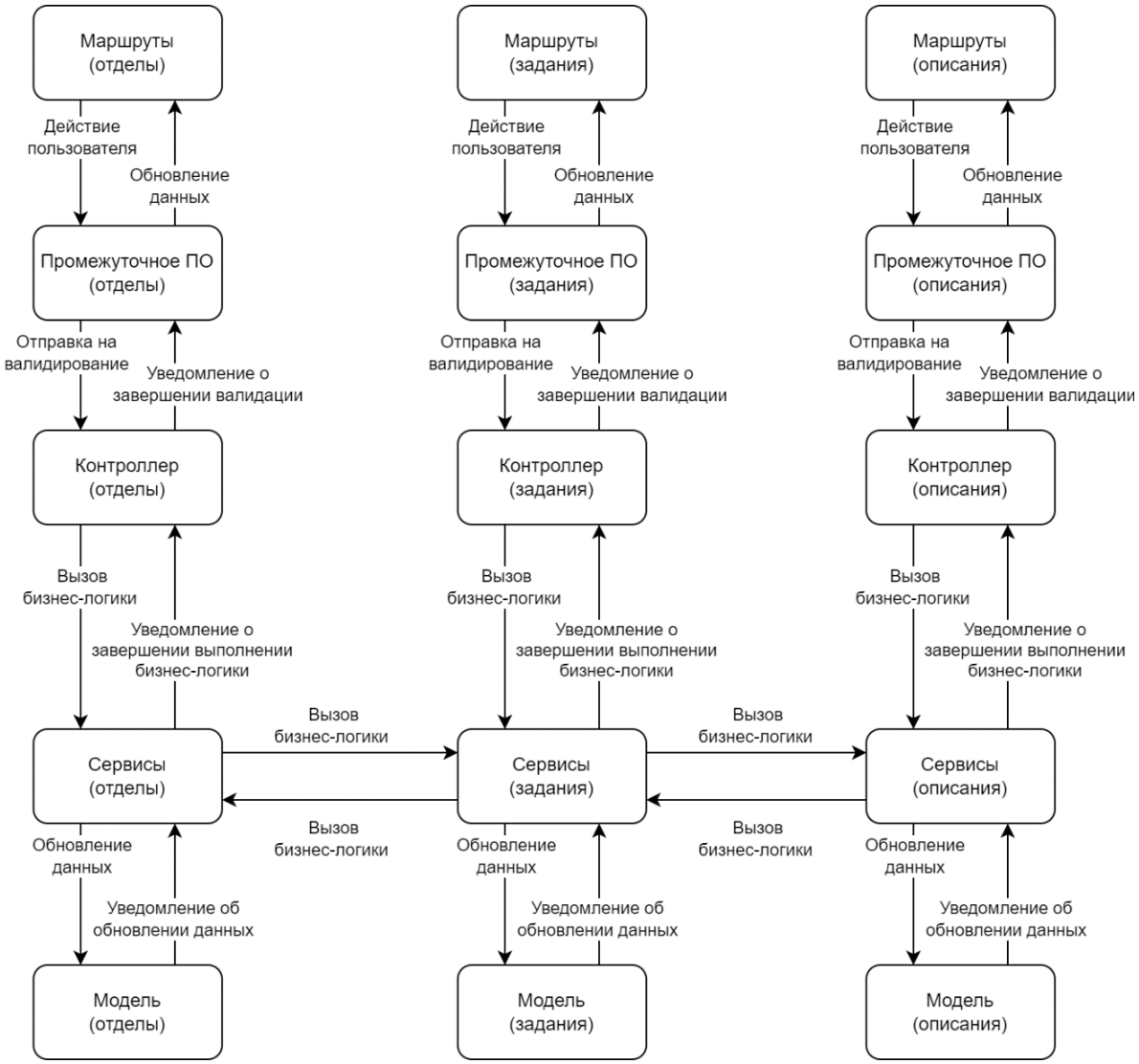

Рис. 5 – Декомпозиция применения шаблона проектирования Модель-Представление-Контроллер-Сервис в разрезе предметной области

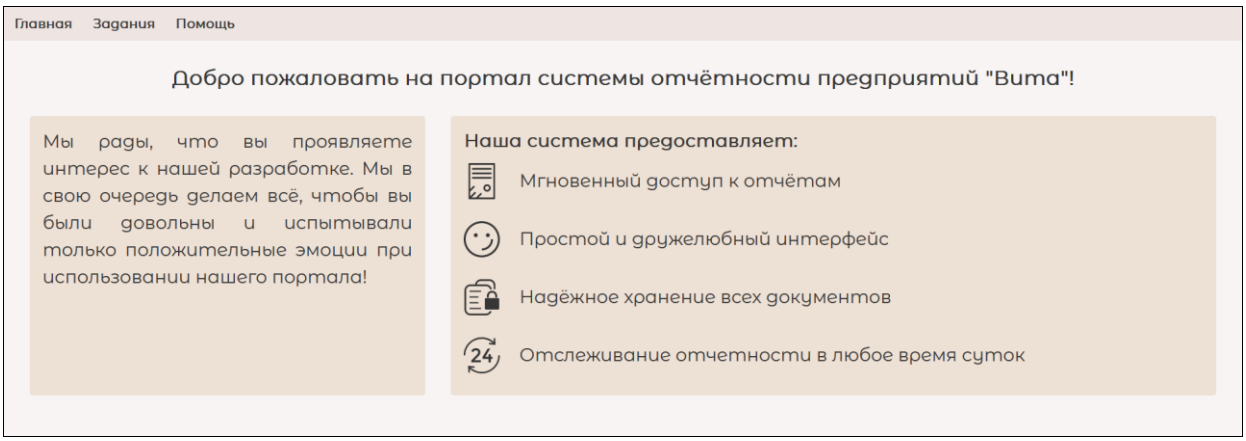

Рис. 6 – Экранная форма реализации решения

Рис. 7 – Экранная форма реализации решения

Рис. 8 – Экранная форма реализации решения

Преимущества использования шаблона проектирования Модель-Представление-Контроллер-Сервис при разработке автоматизированных систем сбора и обработки цифровых отчетных документов для промышленных предприятий.

Разработка автоматизированных систем сбора и обработки цифровых отчетных документов для промышленных предприятий является ресурсоёмким и трудозатратным процессом. Проект такого масштаба затягивает процесс разработки и проектирования на долгое время, однако использование образца проектирования Модель-Представление-Контроллер-Сервис решает ряд проблем, которые могут возникнуть в течении жизненного цикла автоматизированной системы. Можно выделить несколько проблем, которые могут появиться при разработке автоматизированной системы сбора и обработки цифровых отчетных документов для промышленных предприятий без использования архитектурного шаблона Модель-Представление-Контроллер-Сервис:

1. Дублирование кода: без использования шаблона проектирования Модель-Представление-Контроллер-Сервис разработчики могут столкнуться с необходимостью дублирования кода, что может привести к ошибкам и усложнить процесс разработки автоматизированной системы;
2. Отсутствие возможности повторного использования бизнес-логики: без разделения бизнес-логики код становится менее модульным, и использование одних и тех же бизнес-правил в разных частях приложения может стать затруднительным или невозможным;
3. Проблемы с инкапсуляцией бизнес-логики: если бизнес-логика не инкапсулирована, то изменение или обновление логики может вызвать проблемы на других уровнях приложения;
4. Сложность тестирования: без разделения приложения на уровни, тестирование может стать сложным и трудоемким процессом;
5. Снижение читаемости и структурированности кода: без разделения на слои, код может стать менее читаемым, понятным и поддерживаемым, что усложнит обнаружение и устранение ошибок.

Таким образом можно выделить следующие преимущества, которые появляются за счет использование архитектурного образца Модель-Представление-Контроллер-Сервис:

- предотвращение дублирования кода;
- возможность повторного использования бизнес-логики;
- инкапсуляция бизнес-логики;
- упрощение процесса тестирования;
- повышение читаемости и структурированности кода.

Положительное влияние применения шаблона проектирования Модель-Представление-Контроллер-Сервис в автоматизированных системах сбора и обработки цифровых отчетных документов для промышленных предприятий.

Использование шаблона проектирования Модель-Представление-Контроллер-Сервис является полезным в долгосрочной перспективе. Это происходит благодаря следующим преимуществам:

- простоте поддержки системы;
- возможности быстрой смены парадигм между монолитным и микросервисным подходами к архитектуре.

В таких сложных автоматизированных системах как системы сбора и обработки цифровых отчетных документов очень часто может возникать необходимость в доработках, потребность в которых появляется непосредственно во время эксплуатации системы. Это происходит, как правило, из-за получения обратной связи от пользователей автоматизированной системы. Также на этапе эксплуатации системы возможно выявление дополнительных ошибок в программном продукте, которые не были выявлены на этапе тестирования. Таким образом сложные информационно-управляющие системы госкорпораций, интегрированных структур и предприятий промышленности нуждаются в поддержке после ввода в эксплуатацию. Простота поддержки обеспечивает повышение скорости внедрение новых доработок и исправление обнаруженных ошибок. Важно сказать и про возможность быстрой смены парадигм между монолитным и микросервисным подходами. Данное преимущество, как и простота поддержки, обеспечивается именно за счёт конструкции, которая предусмотрена архитектурным образцом Модель-Представление-Контроллер-Сервис. Быстрая смена парадигм делает автоматизированную систему сбора и обработки цифровых отчетных документов гибкой, позволяя осуществлять быстрое масштабирование без дополнительных затрат на исправление исходного кода программы. Такое свойство системы обеспечивается за счёт модульности шаблона Модель-Представление-Контроллер-Сервис.

Заключение

Использование шаблона проектирования Модель-Представление-Контроллер-Сервис позволяет одновременно повысить качество работы и безопасность системы. В настоящее время он является популярным решением, использующимся во многих автоматизированных системах. Популярность данного архитектурного образца обусловлена большой областью его применимости. Как правило, данный шаблон применяется в больших системах, где при разработке возникает необходимость изолирования или инкапсуляции отдельных частей программы для повышения ее надежности и стабильности; потребность в упрощении процесса тестирования и оптимизации его результатов; желание повысить читаемость и структурированность программного кода. Полезность и преимущества шаблона проектирования MVCS включают в себя:

- предотвращение дублирования программного кода позволяет оптимизировать процесс разработки, сократить затраты на поддержку и снизить вероятность ошибок.

• возможность повторного использования бизнес-логики повышает эффективность разработки и снижает затраты времени и ресурсов;

• повышение читаемости и структурированности программного кода упрощает его понимание и поддержку;

• повышение безопасности системы за счет инкапсуляции бизнес-логики и разделения доступа к ней;

• упрощение процесса тестирования за счет более четкого разделения ответственности между компонентами системы;

• обеспечение гибкости и масштабируемости системы за счет возможности быстрой смены ее архитектурной парадигмы;

• простота поддержки системы благодаря разделению ответственности между ее компонентами.

Использование шаблона проектирования Модель-Представление-Контроллер-Сервис особенно важно на промышленных предприятий, где надежность и стабильность автоматизированных систем играют ключевую роль в обеспечении непрерывности и эффективности производственных процессов. При практическом применении данного шаблона при разработке автоматизированной системы сбора и обработки цифровых отчетных документов для промышленных предприятий получилось ускорить общий процесс разработки программного продукта, а также упростить процесс тестирования. Это позволило сэкономить время и ресурсы на создание прототипа автоматизированной системы сбора и обработки цифровых отчётных документов для промышленных предприятий.

Список литературы

1. Паттерны для новичков: MVC vs MVP vs MVVM // Хабр URL: https://habr.com/ru/articles/215605/ (дата обращения: 03.01.2024).

2. MVC MVVM MVP // vc.ru URL: https://vc.ru/dev/886396-mvc-mvvm-mvp (дата обращения: 03.01.2024).

3. Looking at Model-View-Controller in Cocoa // Matt Gallagher Cocoa with Love URL: https://www.cocoawithlove.com/blog/mvc-and-cocoa.html (дата обращения: 08.01.2024).

4. Model-View-Controller // Apple URL: https://clck.ru/3B3Cb7 (дата обращения: 09.01.2024).

5. Isn't Cocoa MVC really MVP? // stackoverflow URL: https://stackoverflow.com/questions/14943678/isnt-cocoa-mvc-really-mvp (дата обращения: 12.01.2024).

6. An Introduction to MVCS Architecture // quantiphi URL: https://quantiphi.com/an-introduction-to-mvcs-architecture/ (дата обращения: 05.01.2024).

7. the difference between model-view-viewmodel and other separated presentation patterns // Microsoft URL: https://learn.microsoft.com/ru-ru/archive/blogs/erwinvandervalk/the-difference-between-model-view-viewmodel-and-other-separated-presentation-patterns (дата обращения: 10.01.2024).

8. Воробьев, А.В. Подход к обнаружению и устранению артефактов пространственных изолиний в приложениях Веб-ГИС / А.В. Воробьев, Г.Р. Воробьева // Компьютерная оптика. – 2023. – Т. 47, № 1. – С. 126-136. – DOI: 10.18287/2412-6179-CO-1127.

9. Rodrigo Aparecido Morbach, Gean davis Breda, Leonardo Mendes Proposal of an architecture for sensor networks monitoring in Open Access Metropolitan Area Networks // Journal of Applied Computing Research, vol. 4, n. 1, p. 49-63, Jan/Jun 2014

10. Xuan et al Cross-domain neurobiology data integration and exploration // BMC Genomics. - 2010, 11(Suppl 3):S6

11. Validating with a Service Layer (C#) // learn.microsoft.com URL: https://goo.su/2LqGD3b (дата обращения: 04.01.2024).

References

1. Patterns for beginners: MVC vs MVP vs MVVM // Habr URL: https://habr.com/ru/articles/215605/ (access date: 01/03/2024).

2. MVC MVVM MVP // vc.ru URL: https://vc.ru/dev/886396-mvc-mvvm-mvp (access date: 01/03/2024).

3. Looking at Model-View-Controller in Cocoa // Matt Gallagher Cocoa with Love URL: https://www.cocoawithlove.com/blog/mvc-and-cocoa.html (access date: 01/08/2024).

4. Model-View-Controller // Apple URL: https://clck.ru/3B3Cb7 (access date: 01/09/2024).

5. Isn't Cocoa MVC really MVP? // stackoverflow URL: https://stackoverflow.com/questions/14943678/isnt-cocoa-mvc-really-mvp (access date: 01/12/2024).

6. An Introduction to MVCS Architecture // quantiphi URL: https://quantiphi.com/an-introduction-to-mvcs-architecture/ (access date: 01/05/2024).

7. the difference between model-view-viewmodel and other separated presentation patterns // Microsoft URL: https://learn.microsoft.com/ru-ru/archive/blogs/erwinvandervalk/the-difference-between-model-view- viewmodel-and-other-separated-presentation-patterns (access date: 01/10/2024).

8. Vorobyov, A.V. An approach to detecting and eliminating spatial contour artifacts in Web GIS applications / A.V. Vorobyov, G.R. Vorobyova // Computer optics. – 2023. – T. 47, No. 1. – P. 126-136. – DOI: 10.18287/2412-6179-CO-1127.

9. Rodrigo Aparecido Morbach, Gean Davis Breda, Leonardo Mendes Proposal of an architecture for sensor networks monitoring in Open Access Metropolitan Area Networks // Journal of Applied Computing Research, vol. 4, n. 1, p. 49-63, Jan/Jun 2014

10. Xuan et al Cross-domain neurobiology data integration and exploration // BMC Genomics. - 2010, 11(Suppl 3):S6

11. Validating with a Service Layer (C#) // learn.microsoft.com URL: https://goo.su/2LqGD3b (access date: 01/04/2024).