УДК 37.025, 004.42, 004.415.2, 159.95

ВЫБОР СТРУКТУРЫ ДЛЯ РАЗРАБОТКИ ПРИЛОЖЕНИЯ РАЗВИТИЕ КОГНИТИВНЫХ ПРОЦЕССОВ ЧЕЛОВЕКА

Ступникова-Долбня А.П., студент гр. ПМИ-221, IV курс, Ступников-Долбня Д.И. студент гр. ПМИ-221, IV курс
Научный руководитель: Каган Е.С., к.т.н., доцент, заведующий кафедрой прикладной математики
Кемеровский государственный университет
г. Кемерово

Разработка программного обеспечения, направленного на развитие когнитивных процессов человека, требует продуманной архитектуры, обеспечивающей масштабируемость, поддерживаемость и разделение ответственности между компонентами системы. В данной статье рассматриваются ключевые этапы эволюции архитектуры прототипа минимально жизнеспособного продукта (MVP) и обосновывается выбор архитектурных решений.

Этапы эволюции архитектуры приложения:

На начальном этапе разработки прототипа MVP приложение имело унифицированную структуру с общим окном для работы с настройками, экраном вывода информации и окном результатов. По мере расширения функциональности возникла необходимость модификации структуры приложения.

В ходе развития приложение было разделено на режимы с отдельными окнами. Однако при планировании дальнейшего масштабирования выявилась проблема: в одном окне предполагалось сосуществование различных режимов, что могло привести к нежелательным связям между компонентами и взаимному влиянию режимов друг на друга. [1]

С ростом числа компонентов отслеживание связей между ними становилось всё более трудоёмким. После добавления второго режима было принято решение о разделении приложения на независимые составные модули. Это позволило обеспечить возможность неограниченного добавления новых модулей; реализовать модули, основанные на различных принципах работы; минимизировать взаимосвязи между компонентами. [1]

Каждый модуль был структурирован на отдельные части, включающие:

- ✓ Главное окно (рис 1);
- ✓ Окно настроек (рис 2);
- ✓ Другие функциональные элементы (рис 3).

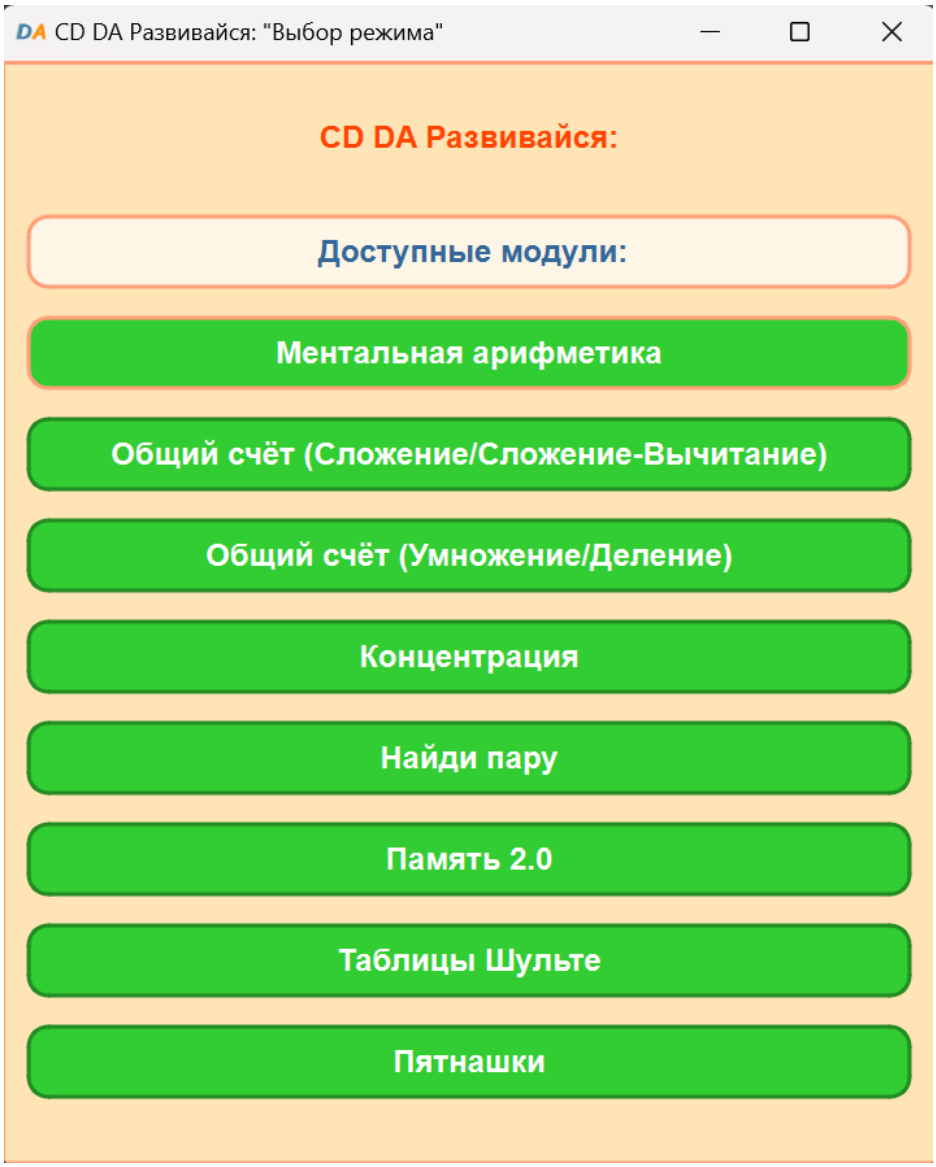

Рис 1. Главное меню модульного приложения

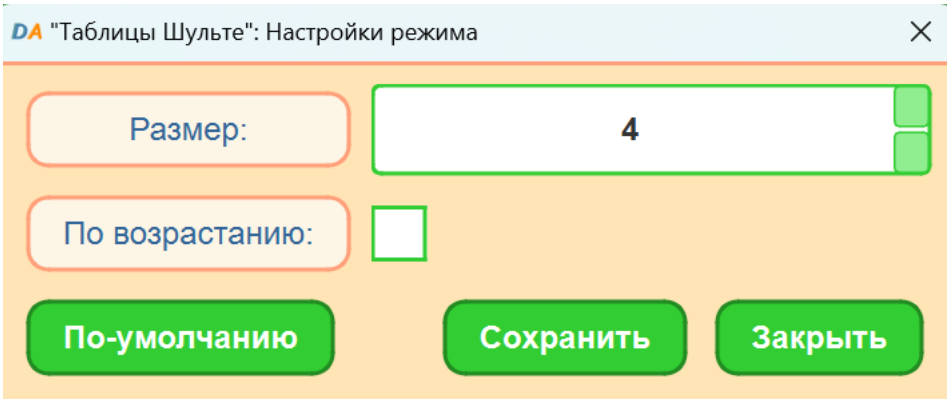

Рис 2. Окно настроек

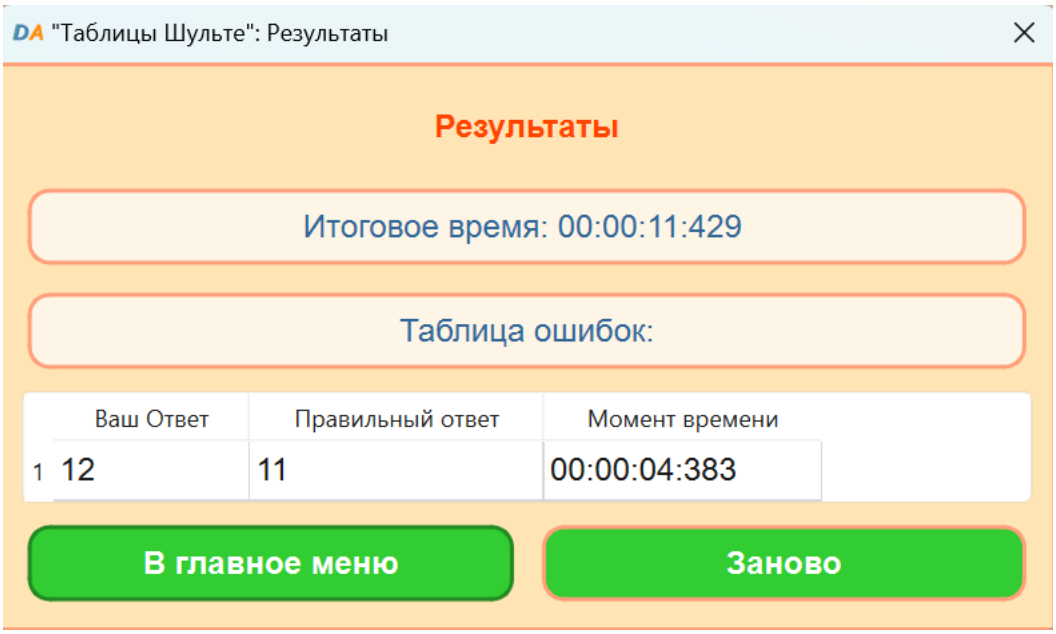

Рис 3. Окно результатов

Такое разделение позволило упростить логику реализации конкретного режима; чётко определить назначение каждого объекта внутри модуля; обеспечить связность объектов посредством общего модуля, выполняющего координирующую функцию.

При дальнейшей разработке обострилась проблема тесной связи между логикой работы окон и их визуальным оформлением. Это приводило к значительным временным затратам на стилизацию отдельных модулей.

Для решения данной проблемы были рассмотрены архитектурные шаблоны проектирования, отделяющие бизнес-логику приложения от пользовательского интерфейса (UI):

MVC (Model-View-Controller) — разделяет приложение на три взаимосвязанных компонента: модель (данные), представление (визуализация) и контроллер (обработка пользовательского ввода).[2]

MVP (Model-View-Presenter) — модифицирует MVC, заменяя контроллер на презентер, который берёт на себя большую часть логики взаимодействия с представлением. [2]

MVVM (Model-View-ViewModel) — использует ViewModel как посредника между моделью и представлением, обеспечивая двустороннюю привязку данных. [2]

Таблица 1. Сравнительный анализ шаблонов

Критерий	MVC	MVP	MVVM
Сложность реализации	Средняя	Средняя	Высокая
Тестируемость	Хорошая	Отличная	Отличная
Гибкость UI	Средняя	Высокая	Очень высокая
Поддержка сложных интерфейсов	Ограниченная	Хорошая	Отличная

В результате анализа для проекта был выбран шаблон MVC благодаря четкому разделению ответственности, простоте реализации, гибкости, а также широкой распространенности и наличию документации. С добавлением новых модулей выявились повторяющиеся функциональные блоки, на основе которых были созданы абстрактные классы, сформировавшие ядро приложения. Ядро включает механизмы работы с UI, компоненты взаимодействия с БД, систему загрузки и инициализации модулей, а также базовые механизмы взаимодействия модулей с менеджерами данных.

Таким образом, реализованные архитектурные решения обеспечили: масштабируемость — добавление новых модулей без существенной переработки существующей кодовой базы; поддерживаемость — упрощение внесения изменений и исправления ошибок за счет четкого разделения ответственности; модульность — независимость компонентов и минимизацию побочных эффектов при модификации; тестируемость — возможность изолированного тестирования модулей и компонентов представленного на рис 1.

Рис 1. Схема архитектуры приложения

Предложенная архитектура обеспечивает устойчивую основу для дальнейшего развития приложения и внедрения новых когнитивных методик. Перспективным направлением дальнейших исследований является оптимизация взаимодействия между модулями и внедрение механизмов динамической загрузки компонентов.

Список литературы:

1. Буч, Г. Объектно-ориентированный анализ и проектирование с примерами приложений / Г. Буч. — 3-е изд. — Москва : Вильямс, 2010. — 720 с. — ISBN 978-5-8459-1651-3. — Текст : непосредственный.

2. Гамма, Э. Приемы объектно-ориентированного проектирования. Паттерны проектирования / Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес. — Санкт-Петербург : Питер, 2020. — 448 с. — ISBN 978-5-4461-1350-7. — Текст : непосредственный.

3. Мартин, Р. Чистая архитектура. Искусство разработки программного обеспечения / Р. Мартин. — Москва : Вильямс, 2018. — 352 с. — ISBN 978-5-907114-01-5. — Текст : непосредственный.

4. Pressman, R. S. Software Engineering: A Practitioner’s Approach / R. S. Pressman. — 9th ed. — New York : McGraw-Hill Education, 2020. — 1056 p. — ISBN 978-1-260-54809-0. — Text : unmediated.

5. Sommerville, I. Software Engineering / I. Sommerville. — 10th ed. — Boston : Pearson, 2016. — 776 p. — ISBN 978-1-292-09613-1. — Text : unmediated.

6. Ampatzoglou, A. Software Architecture Patterns for E-Learning Systems: A Systematic Literature Review / A. Ampatzoglou, I. Stamelos // Journal of Systems and Software. — 2019. — Vol. 157. — P. 110–125. — DOI 10.1016/j.jss.2019.110345. — Text : unmediated.

7. Brown, N. R. Cognitive Training: An Overview of Features and Applications / N. R. Brown, S. L. Thompson, M. A. Smith // Frontiers in Psychology. — 2022. — Vol. 13. — Art. 876543. — DOI 10.3389/fpsyg.2022.876543. — Text : unmediated.

8. Johnson, D. W. Design Patterns in Modern Software Architecture / D. W. Johnson, K. L. Miller // IEEE Software. — 2021. — Vol. 38, no. 4. — P. 67–75. — DOI 10.1109/MS.2021.3078912. — Text : unmediated.

9. Zhang, L. MVC vs. MVP vs. MVVM: A Comparative Study of UI Architecture Patterns / L. Zhang, H. Wang, Y. Chen // International Journal of Software Engineering and Knowledge Engineering. — 2023. — Vol. 33, no. 2. — P. 245–268. — DOI 10.1142/S0218194023500123. — Text : unmediated.