Инженерный вестник Дона, №2 (2022)
ivdon.ru/ru/magazine/archive/n2y2022/7478

Архитектурные приемы при разработке программного обеспечения, зависимого от интерфейса пользователя

О.В. Игнатьева, Н.А. Москат, С.С. Рубцова

Ростовский государственный университет путей сообщения

Аннотация: Рассмотрены архитектурные приемы, использованные в процессе разработки кейсовых заданий для оценки профессиональных навыков обучающихся, реализуемом в рамках учебного процесса РГУПС. Обосновано решение об использовании интерфейса как центра архитектуры. Описаны возможные проблемы в процессе разработки интерфейса указанного программного обеспечения, а также представлены пути их решения. Благодаря представленным приемам построения архитектуры проекта, можно избавиться от основных недостатков, связанных с зависимостью программного кода реализуемого продукта от интерфейса пользователя. Повысить масштабируемость, надежность приложения, сохранив при этом общее качество и функционал.
Ключевые слова: интерфейс, интерфейс пользователя, архитектурное решение, нагруженный интерфейс, программное обеспечение, паттерн, messagebroker, сопрограмма, псевдопоток, кейс.

На пути движения к информационному обществу невозможно обойтись без актуальных программных средств. Одним из ключевых вопросов в процессе разработки программного обеспечения (ПО) является разработка интерфейса. Современный искушенный пользователь в большинстве случаев отдаст предпочтение программному продукту с более понятным и привлекательным интерфейсом, а не только качественно работающему и выполняющему требуемый функционал.

В настоящее время существует целый ряд подходов к построению интерфейсов программных средств: от традиционных [1] до узкопрофильных систем [2]. В зависимости от области применения ПО, требования к интерфейсам также могут весьма разниться. Если речь идет о создании ПО, затрагивающего определенные научные аспекты [3,4], то здесь результат работы программы может быть понятен четкому (заинтересованному) кругу лиц. Следовательно, интерфейс должен быть предельно лаконичен, а иногда достаточно и простого окна (ряда окон) с выдачей полученного результата. Если речь идет о визуальном программировании, например, в процессе

разработки виртуальных тренажеров [5] или симуляторов, то интерфейс становится основным аспектом в общении с пользователем ПО и должен воссоздавать целый ряд особенностей объекта виртуализации. Кроме того, в текущий момент активно развиваются многооконные интерактивные системы, в том числе со стереоскопической визуализацией [6], где за интерфейсом закреплена центровая роль.

Существует тип приложений, опыт пользователя в котором полностью зависит от разработанного интерфейса. Это обусловлено основной выполняемой задачей программного обеспечения. К таким проектам можно отнести системы тестирования и оценки профессиональных навыков и знаний [7]. В приложениях роль интерфейса пользователя, смежная с остальными – это основной посредник между пользователем и алгоритмами приложения. Но именно в системах тестирования и оценки профессиональных навыков и знаний от интерфейса зависит качество выполнения заданий и общий уровень показанных навыков тестируемых пользователей. При неудачной реализации интерфейса даже специалист в IT-сфере будет выполнять задачи медленно и с ошибками. Подход к разработке с интерфейсом пользователя в качестве основного ядра проекта подразумевает отдельный подход к проектированию архитектуры и разработки.

Решения, подразумевающие в своем архитектурном базисе интерфейс, на текущий момент достаточно ограничены и слабо распространены. Это связано с рядом факторов. Прежде всего, с узким кругом типов таких приложений, с устоявшимся мнением о данном подходе к разработке, как о заведомо проигрышном (из-за некоторого количества отрицательных сторон). Но в рамках систем тестирования и оценки навыков, именно решение об использовании интерфейса как центра архитектуры является наиболее удачным. При этом, если адаптировать классические подходы к

проектированию под текущую задачу, то возможно избавиться от минусов в виде плохой масштабируемости, низкой надежности и сильной нагруженности кода.

Для реализации архитектуры на основе интерфейса проекта существует несколько приемов [8]. Некоторые из них можно рассмотреть на практическом примере «Разработка программного приложения ситуационных кейсовых заданий для оценки профессиональных навыков обучающихся», реализуемом в рамках учебного гранта РГУПС. Данное программное приложение предназначено для оценки профессиональных навыков обучающихся по дисциплинам, реализуемым в рамках образовательных программ университета. Обычно подобные классы программного обеспечения формируются с использованием модульной структуры, когда каждый из модулей содержит несколько компонентов, объединенных интерактивными связями [9]. На текущий момент вопросы построения архитектур интеллектуальных систем тестирования весьма актуальны [10].

В рассматриваемом проекте используются комбинации архитектурных решений, которые по сути являются адаптациями некоторых паттернов проектирования, таких, как: Model-View-Controller, Model-View-Presenter, Model-View-View-Model, Observer, MessageBroker и др. При этом реализация, как центральный компонент в системе, использует именно интерфейс пользователя, его части и минимальные элементы.

Первым таким приемом является реализация кнопок, как системы событийных контрактов. Основная проблема в интерфейсе, касающаяся кнопок и их групп – это их теоретически неограниченное количество, а также варьирование количества по мере разработки или поддержки проекта. Также кнопки могут различаться внешним поведением. Например, подсвечиваться при активации. Кроме этого, они относятся к совершенно разным системам,

где могут не являться исполнителями, что гарантирует невозможность их ручной настройки.

В разрабатываемом проекте решением стало использование общего абстрактного функционала, несущего в себе все возможности делегирования и событий. Таким образом, в системы добавляется событие о нажатии (клике) по кнопке через конкретный реализованный класс. При обработке данного события система может вызывать любой функционал, отличный от общего. Каждый класс кнопки является одновременно и контрактом для системы. При этом нельзя назвать такую систему гибкой, так как на каждый контракт в ней необходимо дописывать обработчик.

В качестве решения данной проблемы в реализацию были внедрены универсальные шаблоны, которые позволяют устранить зависимость от созданных ссылочных типов. Такая система похожа на реализацию части контейнера внедрения зависимостей. Но, в отличие от нее, кнопки здесь являются основными элементами. При этом основа посылки сообщений строится на паттерне «Издатель сообщений» (MessageBroker). Данный метод позволяет избавиться от прямой зависимости в отслеживании событий и переложить функционал посылки сообщения конкретным подписчикам на отдельный класс-брокер. Схема работы паттерна приведена на рис. 1.

Рис. 1. – Схема работы паттерна «MesageBroker»

Вторым решением является разработка системы использования внутренних ресурсов через механизм рефлексии.

В системах тестирования и оценки, интерфейс можно назвать нагруженным, так как все задачи и тесты реализуются через него. Это несет дополнительные проблемы с распределением и назначением ресурсов проекта в виде текстов, картинок и других медиаданных. При этом, специфика проекта подразумевает изменение или дополнение ресурсов после релиза приложения. Указанная проблема делает невозможным назначение внутренней базы и ручной настройки всех медиафайлов или данных.

В качестве решения для разрабатываемого проекта ситуационных кейсовых заданий, была реализована система использования ресурсов. В ее основе лежит контейнер, который через список объектов получает ресурсы с жесткого диска и пакует их в структуры проекта. Далее через механизм рефлексии полученные структуры разбиваются на группы ресурсов. Затем, по требованию от системы посылки сообщений, контейнер отдает необходимую структуру. Эта система позволяет добавлять дополнительные данные в уже реализованный проект (например, варианты заданий) путем указания нужных файлов и списков ресурсов.

При этом решение строго зависит от интерфейса пользователя и изменения в нем повлекут за собой необходимость изменений алгоритмов распределения ресурсов.

Общая схема работы системы использования внутренних ресурсов представлена на рис. 2.

Рис. 2. – Схема работы системы использования внутренних ресурсов

Третьим решением является выделение части интерфейса в функционал асинхронных методов сопрограмм. Некоторая часть пользовательского интерфейса должна ожидать ответа или подтверждения. Обычно такие функции реализуют через систему событий. Однако, в разрабатываемом приложении ситуационных кейсовых заданий, это добавит в проект излишние зависимости и дополнительную проблему в виде отсутствия контроля у управленческой сущности над ожидающими окнами. Использование подхода обратной связи (callbacks) также нельзя назвать удачным, так как возможно перенасыщение требований на связь.

Удачным решением стало введение асинхронных связующих методов. Такие методы принадлежат управленческой сущности, находятся под ее контролем и имеют возможность обратной связи в виде возвращаемых значений методов. Такое поведение можно настроить на покадровое ожидание и добиться существенного преимущества над предыдущими методами – это сохранение семантики класса внутри одного метода. То есть,

все взаимодействие с интерфейсом происходит строго в рамках одного метода, что позволит в более удобной форме обрабатывать результаты, которые пользователь не может предоставить мгновенно. Стоит отметить, что такая реализация не является примером многопоточности. Это можно назвать псевдопотоками или сопрограммами, вызываемыми сразу после необходимых работ в главном потоке. При этом, каждый такой псевдопоток непосредственно связан с частью пользовательского интерфейса и соответственно зависим от него. Общая схема работы сопрограммы представлена на рис. 3.

Рис. 3. – Схема работы сопрограммы

Таким образом, в процессе разработки ПО возможно использование различных архитектурных приемов. Благодаря представленным приемам построения архитектуры проекта, можно избавиться от основных недостатков, связанных с зависимостью программного кода реализуемого продукта от интерфейса пользователя, повысить масштабируемость, надежность приложения, сохранив при этом общее качество и функционал. При этом, описанные решения являются только частью из множества возможных. Кроме того, возможны их различные комбинации, в зависимости от сложности и направленности реализуемого приложения.

Литература

1. Лященко З.В., Ведерникова О.Г. Инфраструктура и архитектура информационных систем // «ТрансПромЭк-2019» – Ростов-на-Дону: Ростовский государственный университет путей сообщения, 2019. С. 47-50.

2. Москат Н.А., Алексеенко Е.А. Особенности разработки приложения для расчета и формирования карго-плана // Инженерный вестник Дона, 2019, №4. URL: ivdon.ru/magazine/archive/n4y2019/5880.

3. Chernov A, Butakova M, Guda A, Shevchuk P. Development of intelligent obstacle detection system on railway tracks for yard locomotives using CNN. European Dependable Computing Conference. Springer, Cham. 2020. Pp. 33-43.

4. Ramanathan R. Physical process in Atmospheric Models. MAUSAM. 1993. V. 44. No 1. Pp.125-126.

5. Ведерникова О.Г., Москат Н.А. Расширение и использование редактора визуального программирования для разработки виртуальных тренажеров // Инженерный вестник Дона, 2021, №1. URL: ivdon.ru/ru/magazine/archive/n1y2021/6783.

6. Жигалов К.Ю. Создание и использование интерактивных многооконных ГИС // Современные наукоемкие технологии. 2017. № 6. С. 35-39.

7. Тимофеева М.С., Семенов В.Н. Компьютерное тестирование как инструмент оценки уровня сформированности пула компетенций обучающихся // Автоматизация. Современные технологии, 2018. Т. 72. № 5. С. 232-240.

8. Купер А., Рейман Р., Кронин Д. Интерфейс. Основы проектирования взаимодействия // СПб.: Питер, 2017. 720 с.

9. Кузьмин А.А., Кузьмина Т.А. Архитектура многооконного пользовательского интерфейса электронного учебного пособия // Психолого-педагогические проблемы безопасности человека и общества. – 2020. № 2(47). С. 51-57.

10. Бойко В.А. Архитектура интеллектуальной системы тестирования. // Международный журнал прикладных наук и технологий Integral, 2021. – № 2-1. С. 347-352.

References

1. Lyashchenko Z.V., Vedernikova O.G. Infrastruktura i arkhitektura informatsionnykh sistem [Infrastructure and architecture of information systems]. «TransPromEk-2019» – Rostov-na-Donu: Rostovskiy gosudarstvennyy universitet putey soobshcheniya, 2019. Pp.47-50.

2. Moskat N.A., Alekseenko E.A. Inzhenernyj vestnik Dona, 2019, №3. URL: ivdon.ru/magazine/archive/n4y2019/5880

3. Chernov A, Butakova M, Guda A, Shevchuk P. European Dependable Computing Conference. Springer, Cham. 2020. Pp. 33-43.

4. Ramanathan R. Physical process in Atmospheric Models. MAUSAM. 1993. V. 44. № 1. Pp.125-126.

5. Moskat N.A., Vedernikova O.G. Inzhenernyj vestnik Dona, 2021, №1. URL: ivdon.ru/ru/magazine/archive/n1y2021/6783.

6. Zhigalov K.Yu. Sovremennye naukoemkie tekhnologii. 2017. № 6. Pp.35-39.

7. Timofeeva M.S., Semenov V.N. Avtomatizatsiya. Sovremennye tekhnologii, 2018. T. 72. № 5. Pp.232-240.

8. Kuper A., Rejman R., Kronin D. Interfeys. Osnovy proektirovaniya vzaimodeystviya [About Face. The Essentials of Interraction Desing]. SPb.: Piter, 2017. 720 p.

9. Kuz'min A.A., Kuz'mina T.A. Psikhologo-pedagogicheskie problemy bezopasnosti cheloveka i obshchestva. 2020. № 2(47). Pp.51-57.

10. Boyko V.A. Mezhdunarodnyy zhurnal prikladnykh nauk i tekhnologiy Integral, 2021. – № 2-1 Pp.347-352.