МЕТОД ВНЕДРЕНИЯ МОДУЛЯ АДАПТАЦИИ В ПОЛЬЗОВАТЕЛЬСКИЙ ИНТЕРФЕЙС

УДК 621.391.246

ВЕЛИЧКО Юрий Игоревич
ассистент кафедры информационных технологий Херсонского национального технического университета.
Научные интересы: адаптивные пользовательские интерфейсы.

ВВЕДЕНИЕ И ПОСТАНОВКА ЗАДАЧИ

Создание пользовательского интерфейса (ПИ) давно стало отдельно стоящей задачей в процессе разработки программного обеспечения, решение которой включает в себя не только программирование, но и целый ряд сопутствующих и крайне необходимых процессов, таких как: анализ предметной области функционирования системы, эргономический анализ, графическое прототипирование, тестирование, анализ психологических характеристик потенциальных пользователей, сбор статистических данных о мнении пользователей и другие [1]. В совокупности, разработка качественного ПИ представляет собой сложный, объемный и дорогостоящий, итеративный процесс.

При проектировании ПИ программного продукта (ПП) учитывается множество правил и рекомендаций по построению качественных, дружелюбных интерфейсов. Кроме того, проводится множество исследований направленных на определение целей, предпочтений и методов работы целевой аудитории с разрабатываемой системой. В результате полученный интерфейс отвечает требованиям потенциальных пользователей, и имеет качественный внешний вид. Однако, в большинстве случаев этого оказывается не достаточно, для того, что бы интерфейс обеспечивал максимально возможную продуктивность взаимодействия. Это связано с неудовлетворением индивидуальных потребностей конкретных пользователей[1]. Например:

– размер шрифта может быть слишком мал для людей с плохим зрением.

– подсказки к функциональным элементам интерфейса могут быть слишком детализированными для пользователей с высокой компьютерной грамотностью.

– элемент управления может скрыт за несколькими уровнями иерархии, для пользователей использующих специфические функции системы.

Этот список можно продолжать еще долго, но смысл заключается в том, что современные ПП лишены автоматических модулей адаптации интерфейса к индивидуальным особенностям пользователя. Так же можно констатировать факт, что на сегодняшний день не существует универсальной системы адаптации ПИ, которая могла бы быть применена в различных ПП, функционирующих в разных предметных областях под управлением различных операционных систем.

Универсальная система адаптации интерфейса (САИ) помогла бы решить проблему его несоответствия индивидуальным потребностям пользователя. Для построения такой системы необходимо решить целый ряд задач, одна из которых – метод внедрения данной системы в интерфейс программного продукта. При этом, система адаптации не должна иметь зависимостей или связей с реализацией ПП. Выполнение этого требования поможет разработать систему не зависящую от области применения и способа реализации интерфейса.

Цель работы. Провести анализ основных подходов по проектированию программных продуктов с пользовательским интерфейсом. Выделить основные модели позволяющие создавать гибкие, слабосвязан-

ные системы. Описать технологию внедрения модуля адаптации ПИ в ПП.

ОСНОВНОЙ МАТЕРИАЛ

В процессе развития вычислительной техники, с целью улучшения качества архитектуры программных продуктов, были разработаны 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. Эти конструкции носят название «шаблоны проектирования» или, как принято говорить в профессиональных кругах «паттерны проектирования». Они могут быть реализованы практически на любом языке программирования, претерпевая лишь некоторые модификации. Реализация паттернов зависит от структуры и семантики языка. Назначение паттернов – описать архитектурную конструкцию, представляющую собой решение проблемы проектирования в рамках некоторого часто возникающего контекста[2].

Для разработки программ с пользовательским интерфейсом были разработаны паттерны, позволяющие создавать гибкие и расширяемые системы. Основные из них представлены в табл. 1.

Таблица 1 –

Основные паттерны проектирования программных продуктов с пользовательским интерфейсом

№	Оригинальное название	Аббревиатура	Русскоязычное название
1	Facade		Фасад
2	Model-View-Controller	MVC	Модель-Представление-Контролер
3	Model-View-ViewMode	MVVM	Модель-Представление-«Модель Представления»
4	Model-View-Presenter	MVP	Модель-Представление-Представитель.

Паттерн «Фасад» помогает отделить функциональное ядро систем, так называемый backend, от пользовательского интерфейса – frontend. Суть данного подхода заключается в выделении в рамках системы промежуточных функциональных блоков, которые реализуют обмен данными между компонентами интерфейса и модулями системы. Структура программного продукта, разработанного с использованием шаблона «фасад», представлена на рис. 1.

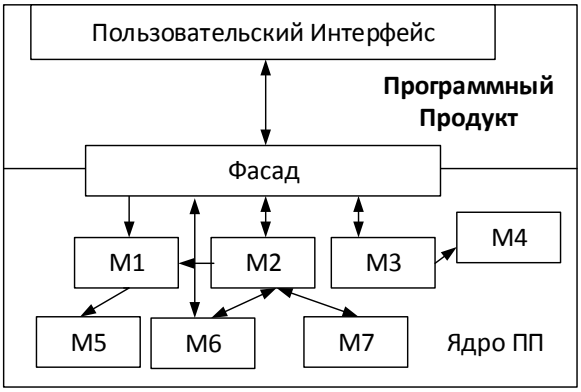

Рисунок 1 – Структура ПП с применением шаблона «Фасад»

Сущность «фасада» заключается в том, чтобы отделить пользовательский интерфейс от ядра системы путем предоставления промежуточного модуля для передачи информации между подсистемами. Фасад не выполняет никаких проверочных действий над данными, которыми он оперирует, и тем более не изменяет их. Также он не имеет возможности воздействовать на интерфейс либо на ядро системы [3]. Можно сказать, что фасад определяет программный интерфейс системы, который в последствии будет отображен пользователю. «Фасад» считается примитивным шаблоном и его не рекомендуется применять при проектировании сложных систем, поскольку он не предоставляет гибкости при расширении, изменении или масштабировании программ из-за высокой связности между функциональными модулями и интерфейсом.

Более гибким решением при проектировании систем с ПИ является применение паттерна модель-представление-котроллер (MVC). Основная цель применения этой концепции состоит в разделении модели данных и обработку событий от её визуализации. Основное преимущество такого подхода заключается в том, что те же самые данные могут быть одновременно представлены в различных контекстах или с различных точек зрения [4]. Кроме того, MVC имеет следующие достоинства:

– Одна модель может быть связана с несколькими представлениями, при этом реализация модели не затрагивается. Например, некоторые данные могут быть одновременно представлены в виде электронной таблицы, гистограммы или круговой диаграммы.

– Не затрагивая реализацию представлений (пользовательского интерфейса), можно изменить обработку действий пользователя (нажатие мышью на кнопке, ввод данных) непосредственно во время функционирования системы (runtime). Для этого достаточно использовать другой контроллер для обработки тех же самых событий.

– При реализации программного продукта, используя данный подход, можно разделить труд разработчиков интерфейса и ядра системы на практически непересекающиеся задачи. Что несомненно повышает производительность работы и сокращает общее время разработки.

MVC является на сегодня негласным стандартом проектирования программных продуктов, включающих в себя пользовательский интерфейс. Остальные паттерны, преимущественно являются его модификацией или адаптацией под конкретную платформу реализации интерфейса. Структура программного продукта разработанная с применением паттерна MVC представлена на рис. 2.

В парадигме MVC ввод пользователя, моделирование внешнего мира, и визуальная обратная связь с пользователем явно отделяются друг от друга и обрабатывается тремя типами объектов, специализированных для выполнения своей задачи. *Представление* управляет графическим выводом в той части экрана, которая отведена приложению. *Контроллер* интерпретирует события, инициируемые пользователем, управляя моделью и (или) представлением, заставляя их изменяться соответствующим образом. *Модель* управляет поведением и данными прикладной области, отвечает на просьбы предоставить информацию о состоянии и отвечает на команды, требующие изменить состояние (обычно из контроллера). Разделение функционирования на эти три задачи — важная концепция, в которой базовое поведение может быть воплощено в абстрактных объектах. Поведение участников MVC затем может наследоваться, добавляться и изменяться по мере необходимости, чтобы обеспечить гибкую и мощную систему [5].

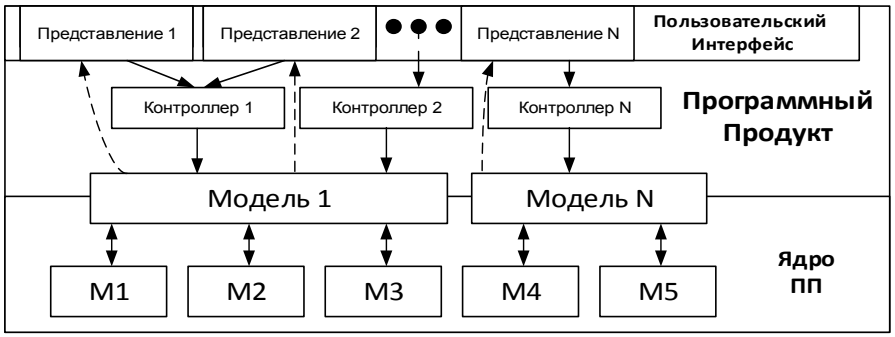

Рисунок 2 – Структура ПП, разработанная с применением паттерна MVC

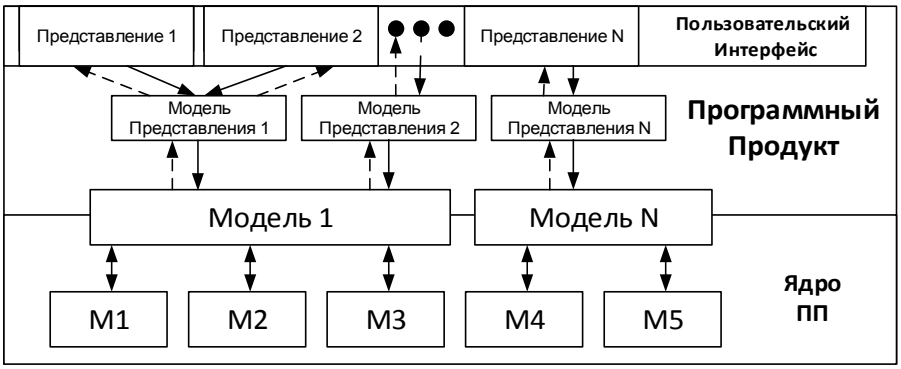

Рисунок 3 – Структура ПП разработанная с применением паттерна MVVM

Шаблон MVC является основополагающей моделью для проектирования приложений с пользовательским интерфейсом. По мере развития информационных технологий, этот подход был расширен и конкретизирован для различного рода приложений. Таким образом был разработан шаблон «модель-представление-«модель представления» или MVVM. [5]. Структура программного продукта построенная с использованием шаблона MVVM представлена на рис. 3.

В отличие от представления в MVC, «модель представления» не нуждается в ссылке на само представление. Представление привязывается к свойствам «модели представления», которая, в свою очередь, представляет данные в объектах модели и других состояниях, нужных для этого представления. Связь между представлением и «моделью представления» устанавливается довольно просто, потому что объект модели представления устанавливается как контекст представления. Если изменяются значения в «модели представления», эти новые значения автоматически переходят в представление через привязку данных. Все изменения данных модели всегда производит «модель представления», а не представление. Классы представления не имеют связей с классом модели, а «модель представления» и модель не связаны с представлением. Модель, в таком случае, вообще не зависит от «модели представления» и представления. Это очень слабо связанная конструкция, что дает ряд преимуществ при проектировании сложных систем включающих в себя интерфейс пользователя.

Еще один из наиболее распространенных шаблонов проектирования носит название Модель-Представление-Презентатор или MVP. Этот метод популярен на различных платформах программирования пользовательских интерфейсов, и представляет собой разновидность шаблона модель-представление-контроллер. В MVP то, что видно на экране — это представление. Данные, которые там отображены — это модель. Презентатор объединяет их вместе [5]. Структура программного продукта с пользовательским интерфейсом, построенного с применением паттерна MVP представлена на рис. 4.

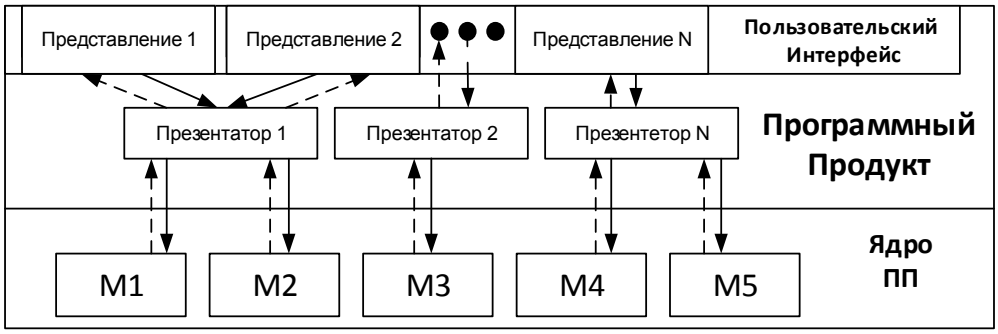

Рисунок 4 – Структура ПП разработанная с применением шаблона MVP

Представление нуждается в презентаторе для заполнения данными модели, реакции на ввод пользователя, предоставления проверки ввода (в том числе за счет передачи этой функции модели) и других подобных задач. Особенностью этого шаблона является то, что он применяется исключительно для реализации программ с пользовательским интерфейсом. Презентатор несет на себе не только функцию отображения модели, но и обработки пользовательских событий.

Основной чертой вышеуказанных методов является отделение реализации логики программного продукта от реализации его интерфейса, что позволяет говорить о разработке интерфейса как отдельного, независимого модуля. Управление интерфейсом производится, как правило, посредством контроллеров, задача которых обрабатывать события от пользователя, передавать их в систему и изменять вид интерфейса в зависимости от состояния системы. Если программный продукт разрабатывается по одному из вышеперечисленных подходов (или их модификации) то в такой ПП можно внедрить систему адаптации интерфейса. Причем, это можно сделать таким образом, что не функциональные модули ПП, не сам интерфейс, не будут зависеть от реализации и функционирования

САИ. При этом ПП так же не будет оказывать никакого влияния САИ не при проектировании, не во время функционирования. Структура ПП использующего САИ представлена на рис. 5.

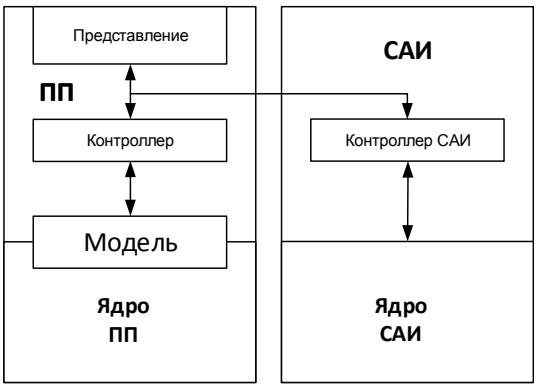

Рисунок 5 – Метод внедрение системы адаптации интерфейса в программный продукт

САИ может быть построена по технологии описанной в паттерне MVC, в которой контроллер является структурой, отвечающей за взаимодействие между интерфейсом программного продукта и ядром системы адаптации интерфейса. Контролер, в самом обобщенном виде, должен выполнять две функции – регистрировать события интерфейса и изменять состояния интерфейса. Ядро должно реализовывать логику адаптации интерфейса основываясь на событиях и формировать сценарии адаптации, которые впоследствии будут реализованы контролерами. Основное преимущество данного метода внедрения САИ в интерфейс ПП состоит в отделении реализации логики адаптации интерфейса от реализации модуля выполняющего изменения интерфейса. Это позволит использовать единую САИ для адаптации интерфейса ПП реализованных на различных платформах. Для этого необходимо, что бы контроллер САИ был включен в ПП и соединялся с ядром по клиент-серверной технологии.

ВЫВОД

Анализ наиболее распространённых методов проектирования программных продуктов с пользовательским интерфейсом показал, что существует возможность внедрить в любой программный продукт модуль адаптации интерфейса, при этом не нарушая и не изменяя структуру самого ПП. Такой подход поможет построить универсальную систему адаптации интерфейса не зависящую от области применения и способа реализации.

ЛИТЕРАТУРА:

1. Malinowski U. et al. A taxonomy of adaptive user interfaces //HCI-92. – 1992. – p.391-414.
2. Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides, *Design Patterns* (Reading, Massachusetts: Addison-Wesley, 1995).
3. Robert C. Martin, Dirk Riehle, Frank Buschmann. Pattern Languages of Program Design 3 (The Software Patterns Series) – Addison-Wesley, 1997. – 672 p.
4. Steve Burbeck. Applications Programming in Smalltalk-80(TM): How to use Model-View-Controller (MVC) http://st-www.cs.illinois.edu/users/smarch/st-docs/mvc.html
5. Martin Fowler. Analysis Patterns: Reusable Object Models – Addison-Wesley, 1997. – 672 p.

Рецензент: *д.т.н., проф. Ходаков В.Е.,*
Херсонский национальный технический университет, Херсон.