Модели создания дружественных графических интерфейсов пользователей в сети интернет литературный обзор

Жукова Александра Владимировна,
студент-магистр
института комплексной безопасности
и специального приборостроения

Авторы работы [1] рассказали о современных шаблонах, которые используются для разработки графических интерфейсов и их функционале, а так же описали их прототипы, подходящие создания для мобильных телефонов. Существует ряд современных мобильных прототипов, которые поддерживают разработку прототипов для мобильных платформ. Большинство этих инструментов не поддерживают использование коллекций шаблонов дизайна пользовательского интерфейса для мобильных пользователей, таких как Библиотека шаблонов материалов для Google. Поддержка предоставляется только для низкоуровневых физических виджетов, таких как кнопки и списки, и их поведение. Эти прототипы предназначены для использования без кода, прототипов и функциональности не могут быть добавлены к ним. Единственная поддерживаемая функциональность — это возможность связывать. Существующая функция средства прототипирования мобильных устройств Proto.io была расширена для поддержки использования выбранных шаблонов и материалов для создания шаблонов прототипов. Парсер был написан для преобразования этих шаблонов прототипов, экспортированных из Proto.io в формате HTML, в Android-совместимый XML, который можно было импортировать в интегрированную среду разработки Android Studio (IDE) для дальнейшей разработки. Было проведено исследование пользователей для сравнения двух разных подходов к созданию мобильного приложения: полного создания мобильного приложения в Android Studio или использования расширенной версии Proto.io для разработки мобильного пользовательского интерфейса и импорта созданного XML в Android Studio для дальнейшего развития. Эти два подхода были сопоставлены с точки зрения эффективности и удовлетворенности пользователей, что оценивалось несколькими дизайнерами. Было установлено, что подход, основанный на использовании мобильного интерфейса пользовательского интерфейса, превосходит подход, ориентированный на развитие.

Например, авторы работы [2] предложили свой вариант создания дружественного графического интерфейса на примере интерактивной обложки для домов престарелых с деменцией. Они поставили себе цель описать сложный процесс проектирования интерактивных работ для жителей домов престарелых, в сотворчестве со всеми заинтересованными сторонами и поделиться используемыми методами и извлеченными уроками. В процесс проектирования были вовлечены домашние медсестры из психиатрического отделения, неформальные воспитатели, представители клиентов, специалисты в области здравоохранения. Этот процесс состоит из трех этапов:

1. определение требований

2. разработка прототипа

3. проведение тестов на удобство использования

Было использовано несколько методов (например, сеансы совместного созыва, «Волшебник из страны Оз»). Каждый этап позволял получить знания и опыт, которые использовались в качестве отправной точки следующего этапа.

В конечном итоге выяснилось, что участники с трудом обращали внимание на установку и интерфейс. Однако, как оказалось, был неиспользованный потенциал для получения захватывающего опыта, сосредоточенного больше на самом содержании интерфейса (например, создание определенных сцен с подсказками для взаимодействия, сцен на основе существующих знаний или предыдущем опыте.) В процессе проектирования были получены «Пятнадцать уроков», которые могут потенциально помочь в разработке интерактивного произведения для домов престарелых, страдающих деменцией.

Этот проект содержит инструменты и передовую практику для заинтересованных сторон, которые позволяют сделать наиболее приемлемый выбор при создании интерактивных работ. Он также иллюстрирует, как совместное проектирование может изменить ситуацию между получением приятного и значимого опыта. Итоги:

· Со-дизайн со всеми заинтересованными сторонами может уменьшить разницу между получением приятного и значимого опыта.

· Есть неиспользованный потенциал для получения впечатляющего опыта, сосредоточенного на самом контенте. Контент как интерфейс оказался важной частью общего пользовательского опыта.

· В тематическом исследовании представлены инструменты и передовая практика (15 «извлеченных уроков») для заинтересованных сторон.

Авторы работы [3] предложили вариацию создания и описания динамических пользовательских интерфейсов для систем управления.

Эта работа представляет собой основу, схожую по сути с предложенным вариантом в работе [2], для создания динамических визуальных интерфейсов для улучшения ситуационной осведомленности. Предлагаемая структура определяет релевантность доступных частей информации, а затем применяет полученные оценки релевантности к визуализации, так что наиболее релевантная и важная информация подчеркивается конечным пользователям. В представленной работе априорное экспертное знание кодируется в системе с использованием Fuzzy Logic (FL). Полученная система вывода FL присваивает оценки фрагментам информации на основе информации состояния системы и определяемой пользователем релевантности. Эти оценки могут затем использоваться для организации и отображения соответствующих данных с учетом текущей ситуации и роли конечных пользователей. Предлагаемая система скоринга FL основана на наборе данных реальной системы управления, и мы демонстрируем, как визуализация информации динамически адаптируется для улучшения ситуационной осведомленности.

Однако такой подход хорош для статичной системы, но что если представить систему в динамичном виде? Авторы работы [25] представили разработку дизайна веб-интерфейса пользователя на основе динамических рекомпонентов. По их мнению для среды и пользователей нужны простые в изменении проблемы в приложении, идеология гибкого программного обеспечения и дизайн веб-интерфейса, объединенные в документе, они предложили возможность динамической реконфигурации гибкой компонентной модели веб-интерфейса на основе компонентов. Модель отображает описание шаблонов стиля компонента и адаптируется к структурным изменениям правил компонентов бизнес-данных, которые хранятся в XML-документах и реляционных базах данных, и позволяет увеличить адаптируемость и облегчить повторное использование веб-интерфейса пользователя. Наконец, гибкий веб-интерфейс пользователя отображения данных таблицы служит для иллюстрации эффективности и доступности модели. Понятие гибкого интерфейса интернет-пользователя

Гибкое программное обеспечение может продолжать работать с системами приложений

и приложениями, изменяя потребности пользователей в определенном диапазоне. Подчеркнутые факторы изменения в гибком программном обеспечении дизайна, который приспосабливается, могут изменять часть функций программного обеспечения, функции напряжения, которые должны быть полностью отражены в пользовательском интерфейсе. Гибкая технология, структура программного обеспечения статическая, централизованная, единственная структура для модульной, децентрализованной, распределенной и автономной структуры преобразования, поддерживающая структурные и функциональные изменения в принципах и методах, таких как скрытие информации, инкапсуляция, разделение и реализация интерфейса, полиморфизм и динамическое связывание, отражение. Исследователь будет внедрять эти принципы и методы в новые языки программирования и методы (объектно-ориентированные языки)проектирования, технологии компонентов программного обеспечения архитектурный стиль, динамическое развитие технологий, программное обеспечение для динамической настройки, рабочий процесс, гибкие отчеты, плагины и технологии двигателей бизнес-правил. Программное обеспечение будет самостоятельно адаптироваться к изменению его возможных механизмов адаптации в рамках программного обеспечения посредством завершения программных манипуляций и взаимодействия с окружающей средой для завершения и изменения поведения в эволюции структуры программного обеспечения.

Заключение

Необходимо резюмировать достоинства и недостатки применения интерфейсов. Достоинствами являются зацепление взаимодействующих частей приложения существенно снижается вплоть до необходимого и достаточного уровня. Посредством использования интерфейсов удается четко разбить задачу на подзадачи, чья реализация может вестись полностью параллельно. Благодаря четко выделенным интерфейсам реальное повторное использование кода существенно возрастает. Отсутствие неконтролируемой привязки прикладного кода к системно-зависимым API существенно упрощает портирование. Благодаря повсеместному применению интерфейсов, проблема безболезненного изменения поведения отдельных частей приложения находит эффективное решение: приложение представляет собой набор четко ограниченных кубиков, взаимодействующих между собой посредством формально заданных интерфейсов. Проектирование приложения посредством создания интерфейсов сразу на целевом языке полностью исключает весь пласт ошибок и чужеродных не эффективностей, возникающих при первоначальном проектировании в рамках «универсальных моделей» типа UML с последующим переводом.

А недостатки в данном случае являются прямым продолжением достоинств. В силу повсеместного использования «дополнительного уровня косвенности» взаимодействие отдельных реализаций интерфейсов (ведь именно они, в конечном итоге, выполняют всю работу) становится достаточно трудно отследить. Простой, написанный «в лоб» код выглядит более очевидным, но эта его простота уже совершенно не подходит для действительно сложных систем, требующих постоянного внесения изменений. Опять же, из-за внесения множества косвенных (фактически, виртуальных) вызовов страдает производительность. Тем не менее, не следует считать, что производительность становится неприемлемой (и именно по этой причине), т.к. необходимость оптимизации «узких мест» свойственна любой сложной системе. Тем более, что дополнительная потребность в ресурсах сравнительно невелика и растет (даже менее чем) линейно относительно числа функций, ставших виртуальными. Как всегда, действительно существенный прирост производительности дает только замена самого алгоритма, а не технических деталей него реализации.