Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Сибирский государственный университет телекоммуникаций и информатики»
(СибГУТИ)

Е. Ю. Мерзлякова

Визуальное программирование и человеко-машинное взаимодействие

Практикум к курсовой работе

Новосибирск 2023

УДК 004.5(075.8)

Утверждено редакционно-издательским советом СибГУТИ

Рецензент *д.т.н.*

Мерзлякова Е. Ю. Визуальное программирование и человеко-машинное взаимодействие/ Е. Ю. Мерзлякова; Сибирский государственный университет телекоммуникаций и информатики; каф. прикладной математики и кибернетики. – Новосибирск, 2023. – 22 с.

Практикум предназначен для студентов технических специальностей, изучающих дисциплину «Основы визуального программирования и человеко-машинного взаимодействия» и содержит методические указания к выполнению курсовой работы.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее время происходит активный рост числа пользователей электронных устройств. Интерфейсы различных приложений используются повсеместно при выполнении разных повседневных задач в обычной жизни, в связи с чем возросла и потребность в грамотно организованном человеко-машинном взаимодействии.

В данной задаче особую роль играет вопрос об оптимизации информационных систем, соблюдении правил построения интерфейсов и учета психологических особенностей пользователей. Эффективный человеко-машинный интерфейс позволит также сократить нагрузку технической поддержки программного обеспечения и снизить риски.

Методические указания к выполнению курсовой работы содержат основную информацию для пошаговой разработки интерфейса. Обучающиеся смогут провести цикл разработки приложения по предложенным вариантам.

1. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

1.1 Проблемно-центрированная разработка интерфейса.

Одним из наиболее эффективных подходов к разработке интерфейса с пользователем, предлагаемых в литературе по человеко-машинному взаимодействию, является подход, сфокусированный на задачах, которые нужно решать пользователю – проблемно-центрированный подход. При таком подходе процесс разработки структурируется исходя из специфических задач, которые пользователь должен будет решать с помощью разрабатываемой системы. Эти задачи выбираются на ранней стадии разработки, затем они используются для выявления требований к дизайну, чтобы облегчить выработку решений и их оценку по мере развития проекта. Основные этапы разработки проблемно-центрированного дизайна:

- анализ задач и пользователей;
- выбор репрезентативных задач;
- заимствование;
- черновое описание дизайна;
- обдумывание дизайна;
- создание прототипа;
- тестирование дизайна с пользователями;
- итерирование;
- реализация;
- отслеживание эксплуатации;
- изменение дизайна.

На этапе **анализа задач и пользователей** предстоит определить, кто и зачем собирается использовать разрабатываемую систему. Осведомлённость об уровне знаний пользователя позволяет дизайнеру ответить на вопросы о выборе названий пунктов меню, о том, какие материалы включить в обучающий комплект и контекстную помощь, и даже о том, какие возможности должна обеспечивать система. Такие различия среди пользователей, как уровень их квалификации, интерес к изучению новых систем, заинтересованность в успехе разработанной системы, могут обусловливать многие решения дизайнера, например, какой заложить уровень обратной связи с пользователем, где предпочесть команды, подаваемые с помощью клавиатуры, а где – выбираемые из меню и т.д. Эффективный анализ задач и пользователей требует персонального контакта между командой разработчиков и теми людьми, которые в дальнейшем будут пользоваться системой.

После выработки хорошего понимания пользователей следует выделить несколько **репрезентативных задач**, которые будут решаться при использовании системы. Это должны быть задачи, которые пользователи описали разработчикам. Первоначально они могут быть описаны буквально в нескольких словах, но, т.к. речь идёт о реальных задачах, в любой момент в дальнейшем эти описания могут быть расширены до любой степени детальности, чтобы ответить на любые вопросы, касающиеся дизайна

интерфейса или анализа предложенных решений. Отобранные задачи должны достаточно полно покрывать всю функциональность системы, полезно сделать проверочный список всех функций и путём сопоставления его с перечнем задач удостоверится, что желаемое покрытие достигнуто.

На этапе **заимствования** необходимо найти существующие интерфейсы, с помощью которых пользователи могут выполнить требуемую работу, и затем строить идеи новой системы на их базе насколько это законно и возможно практически. Например, если они используют электронные таблицы, то, может быть, ваш дизайн должен выглядеть как электронная таблица. Существующую парадигму будет легче изучить и удобнее применять для пользователей, т.к. они уже заранее будут знать, как работает большая часть интерфейса. Копирование известных решений также полезно для низкоуровневых деталей интерфейса, таких как размещение кнопок и названия полей меню. Например, пусть спецификация вашей системы требует, чтобы в какой-то момент могла быть выполнена проверка правописания. Тогда вы должны посмотреть на элементы управления в подсистемах проверки правописания в текстовых редакторах, которые в данный момент используют будущие пользователи вашей системы. Будет лучше, если в вашей системе данный интерфейс будет построен таким же образом.

Черновое описание разрабатываемой вами системы не следует оформлять в виде компьютерной программы (пока), даже если вы умеете пользоваться какими-либо системами автоматизации разработки. Такие системы вынуждают вас прикрепляться к конкретным решениям, которые ещё слишком рано делать. На этой стадии команда разработчиков может проводить множество дискуссий по поводу того, какие возможности должна включать в себя система и как они должны представляться пользователям. Дискуссия должна следовать проблемно-центрированному подходу. То есть, если предлагается введение новой возможности, то необходимо определить, решению какой из репрезентативных задач эта новая возможность будет способствовать. Возможности, которые не способствуют решению любой из задач, как правило, должны отбрасываться. Либо же список репрезентативных задач должен быть дополнен реальной задачей, которая требует этой возможности программы.

Никакая авиационная компания не будет разрабатывать и строить реактивный самолёт без предварительного инженерного анализа, который предсказывает основные технические характеристики. Стоимость строительства и риск неудачи слишком велики. Точно так же стоимость построения законченного пользовательского интерфейса и его тестирование с достаточным количеством пользователей для выявления всех проблем неприемлемо высока. Существует несколько структурных подходов, которые можно использовать, чтобы исследовать сильные и слабые стороны интерфейса до его программного воплощения. Данный этап называется этапом **обдумывания дизайна**. Один из методов состоит в подсчёте количества нажатий клавиш, движений мыши и мыслительных операций (решений), необходимых для выполнения задач, предписанных

разрабатываемой системе. Это позволяет оценить трудоёмкость выполнения задач по времени и выявить задачи, требующие слишком много шагов. Такой метод называется GOMS анализом. Другой метод основан на приёме, названном познавательный сквозной контроль (cognitive walkthrough, CWT), и позволяет находить места в дизайне, где пользователь может делать ошибки. Как и GOMS, CWT анализирует взаимодействия пользователей с интерфейсом при решении отдельных задач.

После этапа обдумывания дизайна по его описанию на бумаге, приходит очередь **создания макета или прототипа интерфейса**, представляющего собой дальнейшую детализацию предстоящей работы, которую можно показать пользователям. Для этого могут использоваться специальные инструменты для создания прототипов, которые позволяют отделить графическую часть интерфейса от логики функционирования системы. На данном этапе не нужно реализовывать систему целиком. Усилия должны быть сосредоточены на частях её интерфейса, нужных для решения репрезентативных задач.

Опыт показывает, что независимо от того, как тщательно был сделан анализ дизайна на предыдущих этапах, существуют проблемы, которые выявляются только при тестировании дизайна с пользователями. **Тестирование** должно проводиться с людьми, чей уровень образования и специальной подготовки примерно соответствует тому, что будет у реальных пользователей системы. Следует попросить пользователей выполнить одну или несколько репрезентативных задач, решение которых поддерживает система. Здесь оказывается эффективным приём "думать вслух". Вы просите пользователя не только делать необходимые действия для решения поставленной задачи, но и говорить, что он делает и о чём размышляет. Эти комментарии необходимо записывать, так как они дают неоценимые данные для улучшения дизайна системы. Информация, собранная при тестировании системы с пользователями, даёт ответ на многие вопросы: сколько времени потребовало выполнение тех или иных действий и задач в целом, какие допускались ошибки, что вызвало затруднение или удивление у пользователя, даже если это и не привело к ошибке. Записанные комментарии пользователя позволяют понять, почему были допущены ошибки. Без комментариев вы только фиксируете сам факт ошибки, но вынуждены потом догадываться (додумывать за пользователя), почему это произошло. При анализе действий пользователя и его комментариев может выясниться, что он мыслит не так, как дизайнер системы. Это позволит внести корректировки, позволяющие приблизить интерфейс системы к её предполагаемому пользователю.

Тестирование с пользователями всегда показывает какие-то проблемы с интерфейсом. Помните, что цель тестирования состоит не в том, чтобы доказать правильность интерфейса, а в том, чтобы улучшить его. Необходимо проанализировать результаты тестирования, соизмеряя стоимость корректировок с серьёзностью возникших проблем, затем доработать интерфейс и протестировать его снова. Серьёзные проблемы могут даже потребовать пересмотра понимания задач и пользователей, т.е. откатить вас на первый этап проблемно-центрированного подхода. Необходимо на каждой

итерации помнить, что различные возможности и особенности интерфейса не самостоятельны. Например, переделывание меню для устранения проблемы, возникшей при выполнении одной задачи, может создать проблемы для других задач. Некоторые из таких взаимовлияний могут быть найдены при анализе без пользователей с помощью CWT или аналогичных приёмов. Другие же не выявятся без тестирования с пользователями. Когда следует прекратить итерации? Если были установлены специфические требования практичности, то итерации прекращаются, как только эти требования выполнены. В противном случае прекращение итераций – это управленческое решение, принимаемое на основе баланса стоимости и полезности дальнейших улучшений против необходимости выхода на рынок с законченным продуктом или сроков предоставления его пользователям.

Ключевой руководящий принцип в программной **реализации** интерфейса состоит в обеспечении возможности его дальнейшего изменения. Постарайтесь предусмотреть некоторую настройку с помощью небольших изменений значений констант и переменных. Например, если вы пишите собственную подпрограмму для реализации специализированного меню, то не закладывайте жёстко в программу такие параметры, как размер, цвет, количество элементов. Постарайтесь также предусмотреть возможность небольших изменений кода, используя ясное разделение на модули. Если доработки в будущем потребуют замены специализированного меню какой-либо более общей функциональной возможностью, доработки кода должны быть тривиальны. Всё это звучит как обычные требования к хорошему стилю программирования и в действительности ими и является. Но это особенно важно для пользовательского интерфейса, который часто занимает более половины кода коммерческого продукта.

Фундаментальный принцип рассматриваемого подхода состоит в том, что команда разработчиков не должна быть изолирована от всей остальной деятельности, связанной с функционированием системы. Если этот принцип уважается, то разработчик должен иметь контакт с пользователями не только в процессе разработки, но и после выпуска системы. **Отслеживание эксплуатации** - это ключевой момент для всех серьёзных организаций, заинтересованных в своём постоянном присутствии на рынке. Один из методов введения разработчиков в контакт с пользователями – организация поочерёдных дежурств на горячей линии связи с потребителями. Другая важная вещь для больших систем – организация собраний групп пользователей и конференций. Такая работа позволяет лучше понять реакцию пользователей на продукт, который они продают. Эта информация в дальнейшем позволяет улучшить описание задач для новой версии продукта и углубить понимание разработчиком предметной области.

На существующем сегодня рынке компьютеров и программ вряд ли существуют продукты, которые поддерживают свою жизнеспособность без регулярных усовершенствований. Независимо от того, насколько удачно система была спроектирована первоначально, с большой вероятностью она будет терять адекватность с течением лет. Меняются задачи, меняются пользователи, приходит время **изменения дизайна**. Приёмы работы меняются

из-за нового оборудования и программных продуктов. Пользователи приобретают новые навыки и ожидаемые реакции. Разработчики должны стоять вровень с этими изменениями, не только отслеживая состояние той рабочей среды, для которой была предназначена их система, но и развитие всего общества, технологий и методов, потребностей. Следующая версия системы должна не только устранять обнаруженные ошибки и недостатки, но и давать новые возможности пользователям.

Практичность (англоязычный термин – usability) – это одна из важнейших характеристик систем, изучению которой посвящено множество исследований. Управленческая деятельность в серьёзной корпорации может потребовать от вас представить различные показатели, которые количественно измеряют практичность. **Требования практичности** – это целевые значения для таких характеристик, как скорость выполнения репрезентативных задач и допустимое количество ошибок. Эти показатели могут использоваться, чтобы мотивировать разработчиков и обосновывать решения по распределению ресурсов. Целевые значения могут быть выбраны так, чтобы побить конкурентов или обеспечить функциональные нужды для хорошо определённых задач. Например, в целях успешной конкуренции от интерфейса может потребоваться не только полнота и удобство для пользователя, но и достижение результатов типа сокращения среднего времени выполнения задач пользователя на 20 % по сравнению с известными аналогами. Типичным примером специальной разработки в области интерфейса, значительно улучшающим скорость работы, является алгоритм T9 для набора текстов на телефонной клавиатуре.

1.2 CWT-анализ интерфейса.

CWT анализ – это формализованный способ представления мыслей и действий людей, когда они пользуются интерфейсом в первый раз. Аббревиатура CWT означает Cognitive Walkthrough (познавательный сквозной контроль). Всё начинается с того, что у вас есть прототип или детальное описание интерфейса, и вы знаете, кто будет пользователем системы. Вы выбираете одну из задач, решение которых интерфейс должен поддерживать, затем формируете полный письменный список действий, необходимых для выполнения задачи при помощи интерфейса. Далее вы пытаетесь рассказать «правдоподобную историю» о каждом действии, которое должен выполнить пользователь. Для этого необходимо давать мотивацию каждого действия пользователя исходя из его предполагаемых знаний и подсказок и реакций интерфейса. Для каждого действия могут обнаруживаться проблемы, мешающие правильному его выполнению. Эти проблемы записываются, но затем вы предполагаете, что они исправлены, и переходите к следующему действию. В результате у вас остаётся список проблем, что является руководством к действию по исправлению (улучшению) интерфейса.

CWT-анализ позволяет обнаружить несколько типов проблем с интерфейсом.

1. Поставить под сомнение ваши первоначальные и не вполне обоснованные предположения о том, как мыслит пользователь.

2. Выявлять элементы управления, которые очевидны для разработчика, но могут быть непонятны пользователю.
3. Выявлять затруднения с надписями и подсказками.
4. Обнаруживать неадекватную обратную связь, что может заставить пользователя сомневаться в результате и повторять всё с начала, хотя всё было сделано правильно.
5. Показывать недостатки в текущем описании интерфейса.

CWT-анализ фокусируется в основном на проблемах, которые пользователи испытывают при первом взаимодействии, не проходя предварительных тренировок. Такая постановка вопроса чрезвычайно важна для некоторых критических систем, таких как банкоматы или терминалы оплаты. Но та же самая ситуация возникает и в сложных программных системах, когда пользователь выполняет какую-либо задачу впервые. Пользователи часто изучают сложные программы постепенно, углубляясь в детали интерфейса по мере возникновения в том необходимости. Поэтому проблемы "первого знакомства" могут возникнуть даже при взаимодействии с давно используемой программой. Если система построена с уважением принципов CWT-анализа, то она позволяет пользователю плавно подниматься с уровня новичка до уровня эксперта.

Многие упускают необходимость сформировать полный и точный список действий для выполнения задачи. То есть они сами точно не знают, как выполнить задачу, и блуждают по интерфейсу, пытаясь отыскать правильную последовательность действий, а потом они оценивают сложность этого блуждания. Иногда, действительно, бывает полезно оценить сложность такого блуждания, но это не метод CWT. В CWT вы должны начинать, имея в руках полный список отдельных элементарных действий, необходимых для выполнения задания. Если в ходе анализа выясняется, что пользователь испытывает затруднения в определении и выполнении одного из действий, то вас интересует не то, как пользователь выйдет из положения, а сам факт того, что проблема возникла и интерфейс нуждается в доработке.

Основные рекомендации по проведению CWT-анализа сводятся к следующему. Вы определили задачу, класс пользователей, интерфейс и корректную последовательность действий. Далее рекомендуется собрать вместе группу разработчиков и других заинтересованных лиц (в учебных целях вы можете действовать в одиночку). И после этого начинается процесс анализа. Вы пытаетесь рассказать историю о том, почему пользователь выбрал бы каждое действие в списке корректных действий. Вы критикуете историю, чтобы сделать её правдоподобной. Рекомендуется держать в уме четыре вопроса:

- Будут ли пользователи пытаться произвести тот или иной эффект, который даёт действие?
- Видят ли пользователи элемент управления (кнопку, меню, переключатель и т.д.) для осуществления действия?
- Если пользователи нашли элемент управления, поймут ли они, что он производит тот эффект, который им нужен?

- После того как действие сделано, будет ли понятен пользователям тот отклик, который они получают, чтобы перейти к следующему действию с уверенностью?

По результатам CWT-анализа необходимо исправить интерфейс. Большинство исправлений будут очевидны. Сложный случай возникает тогда, когда у пользователя вообще нет никаких причин думать, что действие должно быть сделано. Самое верное решение этой проблемы – исключить это действие; пусть система сама выполнит его. Если так не получается, то необходимо перестроить задачу так, чтобы пользователи получали бы логическое побуждение к выполнению "проблемного" действия.

CWT-анализ позволяет выявить много проблем с интерфейсом, но не даёт ответа на вопрос, насколько сложен интерфейс, т.е. сколько времени тратит пользователь на выполнение задачи.

1.3 Анализ GOMS.

GOMS анализ оценивает время работы с интерфейсом обученного пользователя. Даже если интерфейс успешно прошел CWT-анализ, это не означает, что он оптимален с точки зрения трудоёмкости. Если есть несколько альтернативных вариантов построения интерфейса, то анализ GOMS позволяет выбрать тот из них, который требует меньше времени для решения задачи пользователя. В отличие от CWT, GOMS предполагает, что пользователь уже знает интерфейс, т.е. видит его не первый раз.

Аббревиатура GOMS означает Goals, Operations, Methods, Selections (цели, операции, методы, правила выбора). Модель GOMS состоит из описания методов, необходимых для достижения заданных целей. Методы представляются последовательностями шагов, состоящих из операций, которые выполняет пользователь. Метод может быть назван подцелью, т.о. методы образуют иерархическую структуру. Если существует более одного метода для достижения цели, то включаются правила выбора, с помощью которых в зависимости от контекста выбирается один метод.

В терминологии данного вида анализа цель – это то, что мы раньше называли (репрезентативной) задачей, метод – это список действий пользователя, операции – элементарные действия пользователя.

Операции в GOMS – это элементарные действия, которые нельзя разложить на более мелкие. Причём учитываются как внешние действия, т.е. те, что приводят к видимым физическим эффектам, так и внутренние, связанные с мышлением пользователя. Например, действие (шаг метода) "нажать кнопку <button>" представляется в виде последовательности следующих операций:

- визуально определить местонахождение кнопки (мыслительная операция);
- навести на кнопку указатель мыши (внешняя операция);
- щелкнуть кнопкой мыши (внешняя операция).

Рассмотрим анализ интерфейсов для типичной конфигурации персонального компьютера, где в качестве устройств ввода-вывода выступают

монитор, клавиатура и мышь. Практически все интерфейсные взаимодействия в этом случае можно описать следующими операциями:

K – нажатие клавиши;

B – клик кнопкой мыши;

P – наведение указателя мыши;

R – ожидание ответной реакции компьютера;

H – перенос руки с клавиатуры на мышь или наоборот;

D – проведение с помощью мыши прямой линии (например, выделение или прокрутка текста);

M – мыслительная подготовка (к осуществлению одной из перечисленных операций).

Разные пользователи выполняют указанные операции за разное время. Однако, напомним, что GOMS исследует работу опытного пользователя. Многочисленные исследования выявили средние значения времени операций, выполняемых опытными пользователями. Приведём их значения:

K 0.2 с
B 0.2 с
P 1.1 с
H 0.4 с
M 1.35 с

Время выполнения задачи, посчитанное с использованием этих значений, является хорошей средней оценкой сложности интерфейса и действительно работает на практике.

Время ожидания *R* зависит от характера действия, выполняемого компьютером. Речь идёт только о тех задержках, которые связаны с работой интерфейса (например, ожидание открытия нового интерфейсного объекта). При анализе GOMS, как правило, не учитывается время, расходуемое компьютером на выполнение целевой вычислительной функции (например, преобразование одного формата в другой). Такие вычислительные операции относятся уже к функциональному программному обеспечению; скорость их выполнения определяется быстродействием аппаратуры, объёмом обрабатываемых данных и эффективностью алгоритмов, но не связана со сложностью интерфейса. С другой стороны, очень быстрые реакции компьютера, кажущиеся для человека практически мгновенными (например, эхо-печать символа после нажатия клавиши), также не учитываются. Следует учитывать только ожидания, которые сбивают ритм выполнения других операций. Время таких ожиданий можно оценить при работе с реальной программой следующим образом: Если реакция компьютера достаточно быстрая, но, всё же, ощутима, то "стандартное" время *R* рекомендуется установить 0.25 с.

Время операции *D* также может быть разным в зависимости от величины перемещения и других сопутствующих деталей. Его рекомендуется приблизительно оценить при работе с реальным приложением. В качестве "стандартного" значения можно взять 2 с.

Общий алгоритм действий при проведении GOMS анализа:

- Цель разбивается на подцели,
- для каждой подцели расписываются методы,
- методы могут раскрываться далее через внутренние методы
- Формируется конечная последовательность операций
- добавляются мыслительные подготовки *M*.
- Считается время в секундах

Вся последовательность операций разбивается на семантические группы. Например, набор на клавиатуре слова "слон" выполняется четырьмя операциями нажатия на клавиши *KKKK* и рассматривается как отдельная семантическая единица. Перед ней нужно поставить операцию *M*. Действительно, перед тем как напечатать слово, человек должен сформировать его внутренний образ. Это и отражается добавлением операции *M*. Но после того как образ сформирован, слово печатается без дальнейших раздумий между отдельными буквами. Пример другой семантической единицы – открытие меню. Оно состоит из двух операций *PB*. Но прежде чем они могут быть выполнены, нужно решить, какой пункт меню вам нужен и найти, где он находится. Поэтому перед *PB* (но не между *P* и *B*) нужно поставить *M*. Если затем с помощью ещё одной пары операций *PB* в меню выбирается элемент, то *M* ставить не нужно, т.к. идея о том, что нужно выбрать из меню, уже сформировалась у человека ранее.

Наконец, когда вся последовательность операций выписана, мы получаем общее время выполнения (оценка среднего времени) задачи простым суммированием времён отдельных операций. Но это не единственный результат. Проведённый анализ часто позволяет понять, как можно улучшить интерфейс для сокращения времени решения задачи.

Рассмотренных операций может оказаться недостаточно для некоторых интерфейсов. Например, если вы используете сенсорные панели или графические планшеты, то понадобятся специальные, связанные с ними операции. Их временные характеристики можно определить экспериментально или найти в специальной литературе.

ЗАДАНИЕ ПО КУРСОВОМУ ПРОЕКТУ

1. Необходимо выбрать индивидуальную тему из предложенных вариантов:
   1) Электронный фотоальбом.
   2) Учет ремонтной мастерской.
   3) Планировщик задач.
   4) Обучающее приложение для детей дошкольного возраста.
   5) Редактор картинок.
   6) Магазин оргтехники.
   7) Редактор заметок с уведомлениями.
   8) Создание объявлений товаров и услуг.
   9) Редактор целей и достижений.
   10) Интервальное голодание по календарю.
   11) Редактируемый справочник музеев.
   12) Учет посещений компьютерного клуба.
   13) Журнал событий.
   14) Игровой набор тренировок для мозга.
   15) Тамагочи питомцы.
   16) Энциклопедия полезных растений.
   17) Карточная игра.
   18) Дневник питания и тренировок.
   19) Учет заказов швейного ателье.
   20) Обучение счету для школьников.
   21) Администратор фитнес клуба.
   22) Календарь дней рождений.
   23) Планировщик дел ребенка в картинках.
   24) Помощник пациента с напоминаниями.
   25) Школьный планер.
   26) Игровая ферма.
   27) Создание собственного журнала комиксов.
   28) Система учета кадров агентства по трудоустройству.
   29) Работа кинотеатра.
   30) Учет занятий спортивной школы.

2. Выполнить этапы разработки интерфейса:
   1) анализ задач и пользователей: провести обзор предметной области по выбранной теме, определить круг пользователей и требования к функционалу;
   2) выбор репрезентативных задач: перечислить задачи, которые будет выполнять пользователь вашей программы;
   3) заимствование: проанализировать существующие интерфейсы, которые могут выполнять похожие функции по вашей теме и выбрать лучшие интерфейсные решения;

4) создание прототипа: с помощью любых доступных средств (например, https://disk.yandex.ru/d/UxCXEwMRcOyRWw) построить прототип интерфейса;

3. Реализовать в среде Qt Creator многооконное приложение для выполнения прикладных задач пользователя в соответствии с вариантом. Провести анализ разработанного интерфейса.

Требования:	«удовлетворительно»	«хорошо»	«отлично»
реализация основных задач пользователя по выбранной теме	+	+	+
работа с файлами	+	+	+
использование SQLite	+	+	+
проверки на ввод некорректных значений в программе	+	+	+
CWT,GOMS	+	+	+
использование таймера	–	+	+
информационная наполненность	–	+	+
сохранение результатов предыдущей работы пользователя	–	+	+
использование менеджеров компоновки на всех формах	–	+	+
использование QSettings	–	–	+
реализация навигатора справки	–	–	+
спользование отдельного файла css для всего проекта	–	–	+

ОБРАЗЕЦ ОТЧЕТА

Министерство цифрового развития, связи и массовых коммуникаций РФ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Сибирский государственный университет телекоммуникаций и информатики»
(СибГУТИ)

КУРСОВАЯ РАБОТА

Вариант
ТЕМА РАБОТЫ

Выполнил студент группы группа ФИО

Проверил доцент кафедры ПМиК к.т.н Мерзлякова Е.Ю.

Новосибирск 2023

СОДЕРЖАНИЕ

В содержание включают номера и наименования разделов и подразделов с указанием номеров листов (страниц). В содержание также включают все приложения, вошедшие в данный документ, с указанием номера листа (страницы).

ВВЕДЕНИЕ

Обзор предметной области по заданной теме. Почему выбрали данную тему. Какие существуют приложения по данной либо схожей тематике.

1. Анализ задач и пользователей

Пример:
Для варианта задания «Планировщик задач для управляющего проектами» при рассмотрении заинтересованных лиц были выбраны люди, которые имеют опыт в управлении и командной разработке.
Из данной категории был одногруппник, который занимается командным написанием игрового проекта, и отец, у которого возникают различные задачи в процессе управления работой магазина:
1. Одногруппник: возраст – 20 лет, образование – среднее общее, студент, навыки в выбранной сфере – опыт работы в команде, навыки владения компьютером – высокие.
2. Отец: возраст – 43 года, образование – высшее педагогическое, директор магазина, навыки в выбранной сфере – большой опыт календарного планирования, навыки владения компьютером – средние.
Пользователи программного продукта для планирования задач проекта в большинстве своем являются лицами мужского пола молодого и среднего возраста, имеют незаконченное или полное высшее образование и определенный опыт работы в команде. Навык владения компьютером обычно на уровне базового. Пользователи из рассматриваемой области часто обладают лидерскими качествами и ожидают от приложения простой и четкой помощи в решении своих задач. Существенное значение имеет наглядность процесса управления.
Основные термины пользователя: перечислить.
Возможности существующих программ по схожей тематике: перечислить функции тех приложений, о которых писали во введении.

2. Выбор репрезентативных задач

Пример:

1) Создать карточку путешественника, в которой будет находиться копия паспорта, электронный билет, заметки.
2) Указать название поездки, страну, даты, транспорт, маршрут, список вещей.
3) Добавить вещи на своё усмотрение, а также отмечать, что взял, не взял, и что необходимо купить.
4) Редактирование или удаление поездок
5) Поиск поездок по любому полю
6) Сохранение в отдельный файл карточки путешественника
7) Сохранение в отдельный файл плана поездки
8) Прикрепление фотографий к поездке.

3. Заимствование

Пример

Tripit

Приложение для планирования путешествий. Основной функционал и дизайн будет позаимствован именно с данного приложения.

Например, при заходе в приложение будет отображаться примерно следующее (рис.1):

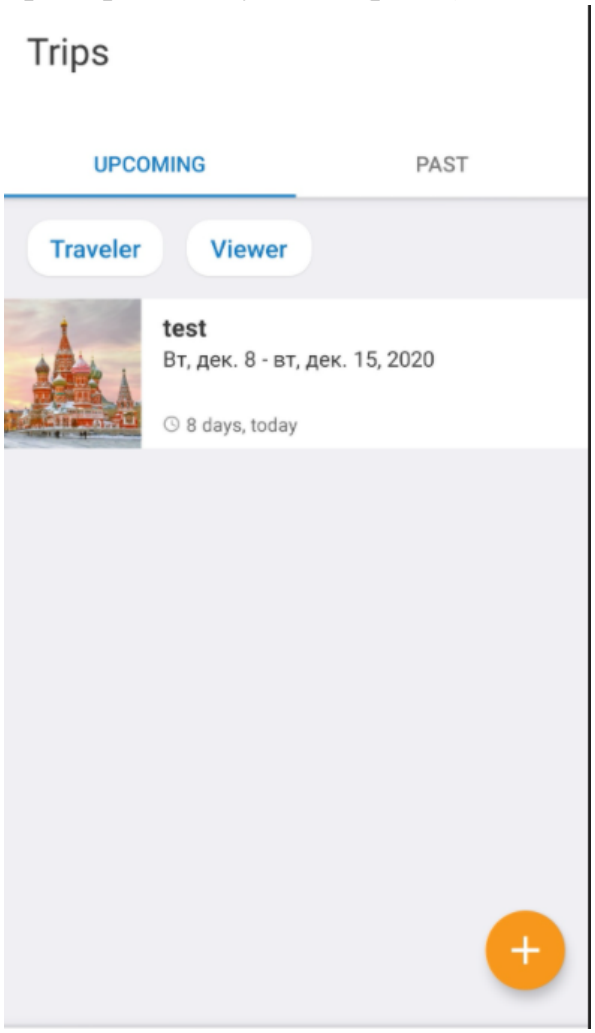

Рисунок 1. Первое окно приложения Tripit

То есть в моем приложении тоже будут картинки, название поездок и даты. Вверху над поездками будет отображаться кнопка добавления поездки, как на рис.2:

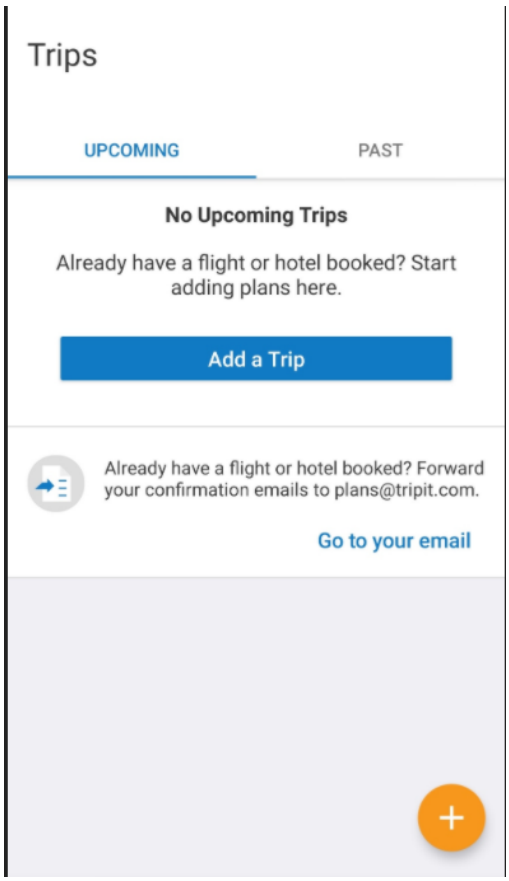

Рисунок 2. Добавление поездки в приложении Tripit

Также заимствовано и само меню создания поездки (в моём приложении будет больше пунктов, в соответствии с репрезентативной задачей) (рис.3):

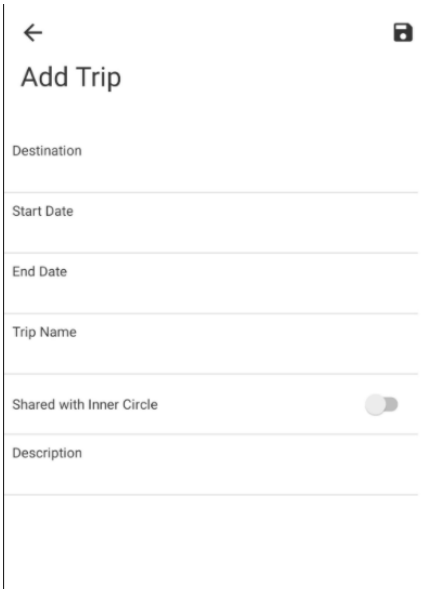

Рисунок 3. Меню в приложении Tripit

В меню будут на видное место добавлены кнопки сохранения и отмены. После того, как пользователь зайдёт в созданную поездку, ему будут доступны остальные функции (список вещей, создание карточки, плана и т.д). Выглядеть это будет подобно рис. 4:

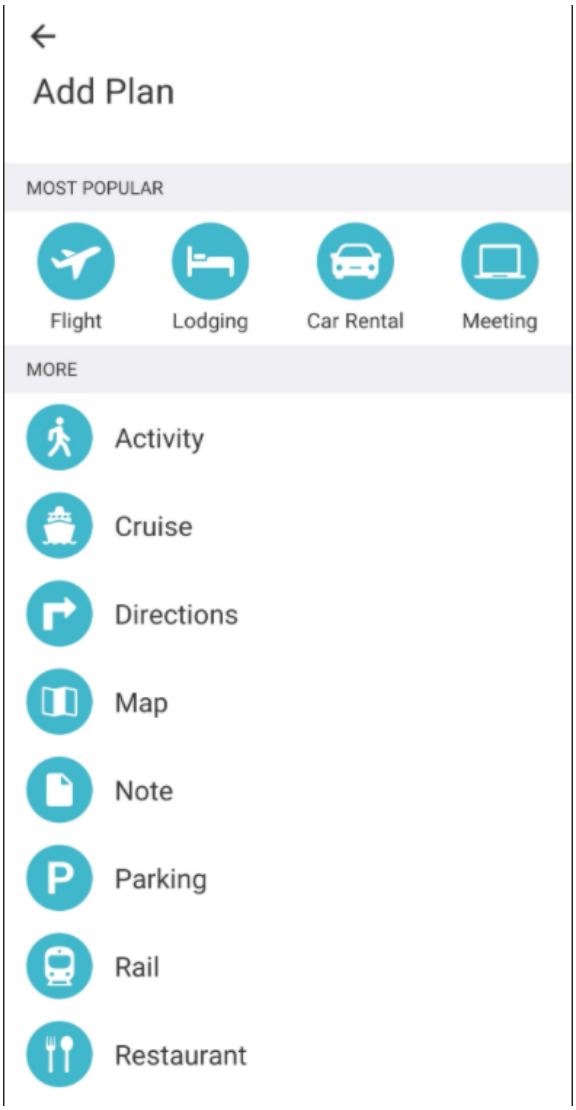

Рисунок 4. Функции приложения Tripit

Пункты данного меню показаны как пример, но в моём приложения пункты будут свои, в соответствии с заданием. Не будет панели с наиболее популярными пунктами, будет список с возможными действиями.

PackPoint

Из данного приложения будет заимствовано меню добавления и выбора вещей, а также указание их количества (рис. 5):

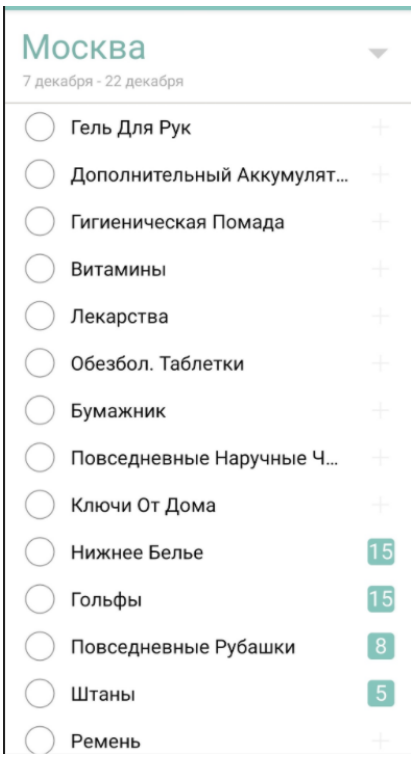

Рисунок 5. Список вещей в приложении PackPoint

В моём приложении также будет пункт «Необходимо купить» на против недостающих вещей.

MyLifeOrganized

Приложение - планировщик задач. Из данного приложения был заимствован способ отображения задач/подзадач в виде древовидной иерархии, в которой каждой задаче/подзадаче в соответствие ставится дата начала/окончания и процент прогресса задачи/подзадачи (рис.6). Также был частично заимствован интерфейс.

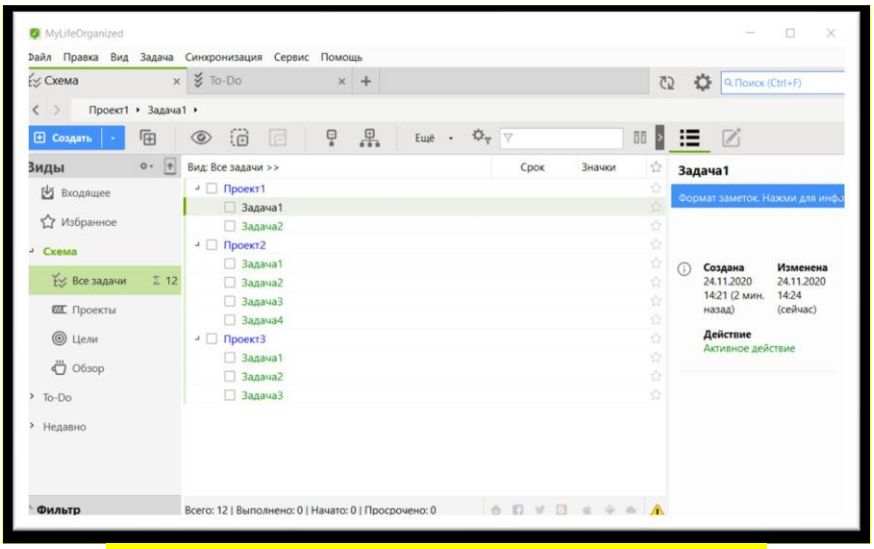

Рисунок 6. Приложение MyLifeOrganized

TickTick

Из данного приложения - планировщика задач, была позаимствована как система приоритетов, так и удобный механизм смены приоритетов в виде выпадающего списка (рис.7).

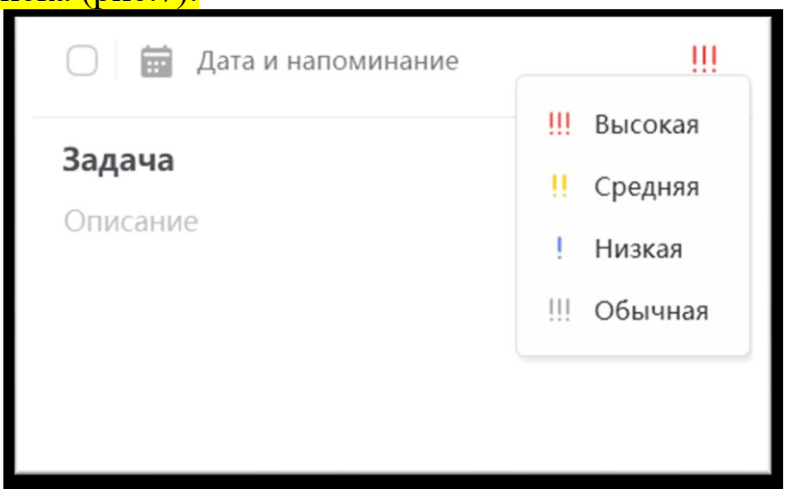

Рисунок 7. Приложение TickTick

4. Прототип интерфейса

пример

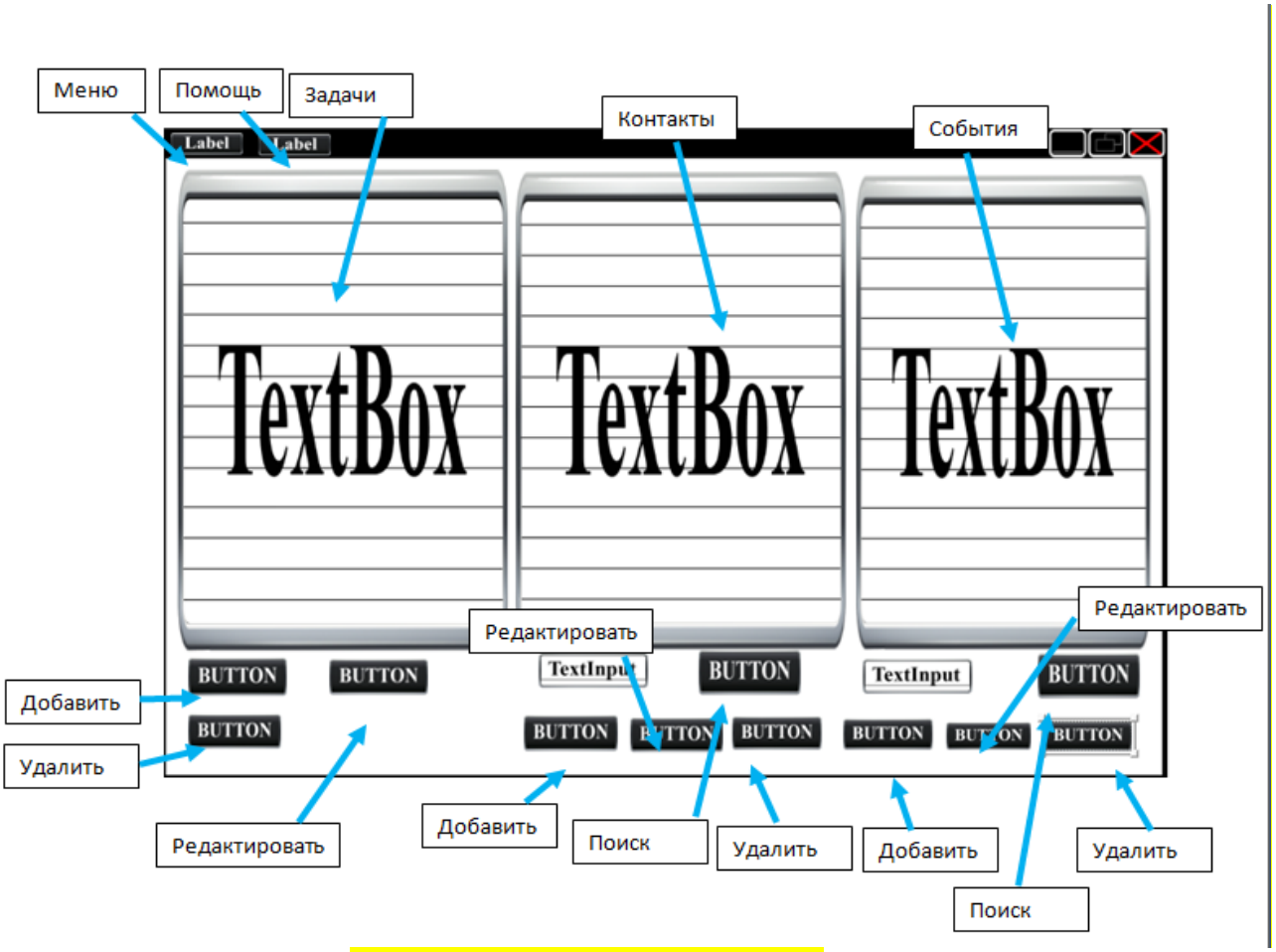

Рисунок 8. Вид главного окна

Описание элементов интерфейса:
Меню: меню будет содержать следующие подпункты

- Добавить задачу: позволяет добавить задачу.
- Добавить контакт: позволяет добавить контакт.
- Добавить событие: позволяет добавить событие.

Помощь: открывает помощь, где подробно описано как пользоваться программой.

Кнопки добавить и удалить под таблицами: позволяют добавить и удалять определённые категории данных, а именно события, контакты и задачи.

Кнопки поиска позволяют искать по названиям в определённых таблицах.

Кнопки редактировать, позволяют редактировать определённые данные в таблицах.

Задачи: данная таблица будет содержать следующие данные: Название задачи, дата выполнения и выполнена ли она или нет.
Контакты: данная таблица будет содержать следующие данные: Имя контакта, контактный телефон, так называемый комментарий, в который можно записать особенности контакта, например, время, когда можно ему позвонить.
События: данная таблица будет содержать следующие данные: Название события, дата проведения и статус проведено, не проведено или отменено.

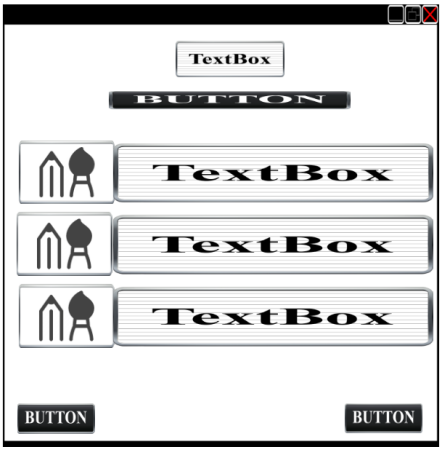

Рисунок 9. Меню приложения

- Данное меню будет открываться при **запуске приложения**. Идём сверху вниз: в первом textbox отображается название меню, в котором сейчас находится пользователь (оно будет во всех меню). Далее кнопка, по нажатию на которой пользователь попадёт в меню создания поездки. После идут уже созданные поездки с фотографией, описанием и датами. В самом низу кнопка справа – выход, кнопка слева – справочная информация.

- На рисунке 10 представлено меню **создания поездки**. Ниже названия меню располагаются поля для ввода информации (название, страна и т.д). Начальную и конечную даты пользователь будет отмечать на календаре, и эти даты будут заноситься в поля ввода. Календарь появится в отдельном окне при нажатии на поле ввода даты. Правая кнопка внизу заменится на «отмену», то есть при её нажатии пользователь выйдет обратно в меню поездок.

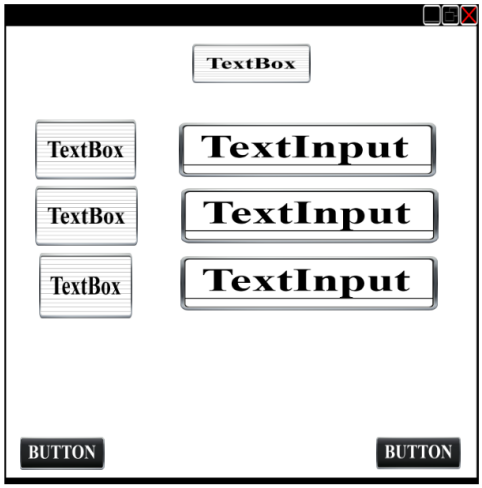

Рисунок 10. Меню создания поездки

- **Меню создания вещей** будет выглядеть следующим образом. Под названием меню располагается краткое описание поездки (название, город, даты). Далее располагается список вещей с возможностью отметить взятие вещи. Справа от вещей находится пункт «купить», нажатие на который убирает отметку о взятии вещи, если эта отметка была. Под списком находится кнопка «Добавить вещь». Меню добавления вещи будет таким же, как в меню создания поездки, только будут другие поля для заполнения.

- **Создание плана поездки и карточки путешественника** будут также запускаться из меню поездки и будут выглядеть также, как и меню создания поездки.

5. Реализация

Функционал: перечислить какие задачи реализованы в вашем приложении.

Приложение состоит из окон: перечислить названия окон, классы.

Привести скриншоты основных окон приложения.

Описать базу данных при ее наличии.

Провести анализ двух задач из пункта 2 (репрезентативные задачи):

- CWT
- GOMS.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

Перечислить источники, которыми фактически пользовались, включая ссылки на анализируемые приложения.

ПРИЛОЖЕНИЕ

- Часть кода программы, где реализуется главный функционал приложения, с комментариями к коду.
- Содержимое файла css (при его наличии)

КОНЕЦ ОБРАЗЦА

СПИСОК ЛИТЕРАТУРЫ

1. Алексеев Е.Р., Злобин Г.Г., Костюк Д.А. [и др.] Программирование на языке C++ в среде Qt Creator : Курс лекций. — Москва : Интуит НОУ, 2016. — 715 с. [Электронный ресурс]. URL: https://book.ru/book/918128 (дата обращения: 05.05.2023).
2. Зайдуллин С. С., Человеко-машинное взаимодействие в информационных системах : учебное пособие — Казань : КНИТУ-КАИ, 2020. — ISBN 978-5-7579-2495-3. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/264929 (дата обращения: 21.09.2023). — Режим доступа: для авториз. пользователей. — С. 5.).
3. Зубкова Т.М. Технология разработки программного обеспечения: учебное пособие. Оренбургский гос. ун-т.- Оренбург: ОГУ, 2017. – 468 с.
4. Корнипаев И.Требования для программного обеспечения: рекомендации по сбору и документированию. М.: Издательство «Книга по Требованию», 2013. – 118 с.
5. Мерзлякова Е. Ю. Разработка прикладных программ в Qt Creator на C++ / Сибирский государственный университет телекоммуникаций и информатики. Новосибирск, 2023. 43 с.
6. Мерзлякова Е. Ю. Визуальное программирование и человеко-машинное взаимодействие : практикум / Сибирский государственный университет телекоммуникаций и информатики. Новосибирск, 2022. 49 с.
7. Шлее М. Qt 5.10. Профессиональное программирование на C++. БХВ-Петербург, 2018. 1052 с.
8. Qt, Tools for Each Stage of Software Development Lifecycle [Электронный ресурс]. URL: https://www.qt.io (дата обращения: 5.05.2023).

Учебное издание

Екатерина Юрьевна Мерзлякова

Визуальное программирование и человеко-машинное взаимодействие

Редактор
Корректор

Подписано в печать 01.01.2023.
Формат бумаги 62 × 84/16, отпечатано на ризографе, шрифт № 10,
п. л. – 2,3, заказ № , тираж – 50.

Отдел рекламы и PR СибГУТИ
630102, г. Новосибирск, ул. Кирова, 86, офис 107, тел. (383) 269-83-18