**Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет «Московский институт электронной техники»**

**Кафедра** информатики и программного обеспечения вычислительных систем

**МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ**

**на тему:**

Разработка методики и алгоритмов проектирования человеко-машинных интерфейсов промышленных систем

на соискание степени магистра по направлению подготовки

09.04.04 “Программная инженерия”

Программа “Программное обеспечение автоматизированных систем и вычислительных комплексов”

**Научный руководитель** д.т.н. профессор

Гагарина Лариса Геннадьевна

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Магистрант** ИПОВС-22

Бычков Андрей Алексеевич

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Москва 2018 г.**

Оглавление

[Введение 3](#_Toc514916096)

[Глава 1. Аналитический обзор современных методов и средств проектирования человеко-машинных интерфейсов 5](#_Toc514916097)

[1.1. Принципы проектирования современных человеко-машинных интерфейсов (ЧМИ) 5](#_Toc514916098)

[1.2. Особенности существующих методов проектирования 7](#_Toc514916099)

[1.3. Существующие средства проектирования 20](#_Toc514916100)

[1.4. Анализ преимуществ и недостатков современных ЧМИ 22](#_Toc514916101)

[1.5. Постановка задачи диссертационного исследования 23](#_Toc514916102)

[Выводы по главе I 24](#_Toc514916103)

[Глава 2. Формализация задачи оптимизации эргономики ЧМИ 25](#_Toc514916104)

[2.1. Анализ эргономических требований 25](#_Toc514916105)

[2.1.1. Ограничения мыслительной деятельности 25](#_Toc514916106)

[2.1.2. Анализ модели памяти человека 27](#_Toc514916107)

[2.1.3. Внешний вид и расположение элементов управления 32](#_Toc514916108)

[2.1.4. Принципы описания «юзабилити» 34](#_Toc514916109)

[2.1.5. Существующие способы оценки эргономики 36](#_Toc514916110)

[2.1.6. Технологии для людей с ограниченными возможностями 37](#_Toc514916111)

[2.2. Математическая модель численной оценки эргономических качеств на основании представленных критериев 39](#_Toc514916112)

[Выводы по главе II 42](#_Toc514916113)

[Глава 3. Разработка модифицированной методики проектирования ЧМИ 43](#_Toc514916114)

[3.1. Разработка UML-диаграмм для определения набора команд, доступных оператору 43](#_Toc514916115)

[3.2. Разработка UML-диаграмм последовательности действий для определения порядка следования элементов управления в макете 46](#_Toc514916116)

[3.3. Разработка алгоритма выбора навигационной парадигмы 49](#_Toc514916117)

[3.4. Разработка алгоритма выбора элементов управления 52](#_Toc514916118)

[3.5. Разработка генетического алгоритма компоновки операторской панели с целью минимизации функции штрафа. 56](#_Toc514916119)

[Выводы по главе III 69](#_Toc514916120)

[Глава 4. Верификация алгоритма компоновки операторской панели 70](#_Toc514916121)

[4.1. Анализ вариантов использования консольных программ \*nix-подобных систем 70](#_Toc514916122)

[4.2. Программная реализация алгоритма компоновки операторской панели 70](#_Toc514916123)

[4.2.1. Оценка существующих программ на основе графической среды 70](#_Toc514916124)

[4.2.2. Разработка верификатора компоновки ЧМИ с применением паттерна MVVM 72](#_Toc514916125)

[4.2.3. Ускорение процесса компоновки с помощью параллельных SIMD вычислений 72](#_Toc514916126)

[4.2.4. Минимаксная оценка скорости сходимости генетического алгоритма в зависимости от алгоритмов селекции и скрещивания 79](#_Toc514916127)

[4.2.5. Визуализация результатов средствами MATLAB 80](#_Toc514916128)

[Выводы по главе IV 81](#_Toc514916129)

[Заключение 81](#_Toc514916130)

[Список литературы 83](#_Toc514916131)

[Приложение 87](#_Toc514916132)

Введение

В настоящее время в области проектирования внешне привлекательных и эргономичных человеко-машинных интерфейсов (ЧМИ) промышленных систем основной проблемой является высокий уровень сложности, приводящий к повышенным затратам на освоение персоналом. Причины сложившейся ситуации заключаются в следующем:

* при проектировании интерфейсов промышленных систем в роли пользователя выступает сам разработчик;
* широкое распространение потребительской электроники с ЧМИ, построенными по отличным от промышленных методикам;
* наличие различных категорий пользователей;
* отсутствие обратной связи от конечных пользователей;
* высокая сложность процессов, производимых промышленным оборудованием.

Очень часто при разработке автоматического оборудования тщательная проработка интерфейсной составляющей относится на второй план, ввиду более приоритетной, с точки зрения производителя, аппаратной или механической части изделия. Использование непродуманного ЧМИ приводит к травмам при производстве. По данным [1] травмы при использовании оборудования происходят по следующим причинам:

* 68% – неправильное взаимодействие с ЧМИ;
* 8% – использование приборов без защитных кожухов;
* 17% – дефекты в оборудовании;
* 7% – неправомерные изменения в конструкции оборудования.

Традиционно считается, что промышленное оборудование предназначено для узкоспециализированных технологических операций, из чего вытекают высокие требованию к операторам данных изделий. Из вышесказанного часто делается ошибочное предположение об отсутствии необходимости в удобстве и эргономике пользовательских интерфейсов данных приборов.

На данный момент известны следующие подходы к оценке эргономики человеко-машинных интерфейсов и существуют программные средства проектирования: GOMS [2], «Теория деятельности» [3], «Ориентированное на пользователя проектирование» [4], и «Value sensitive design» [5]. Однако каждый из этих подходов концентрирует внимание на отдельных аспектах эргономики, и, зачастую, оставляют отдельные моменты непроработанными.

Обилие новых видов средств взаимодействия, таких как сенсорные дисплеи, шлемы виртуальной и дополненной реальности и т.д. ведут к пересмотру уже существующих систем оценки эргономики и обновляют процесс проектирования. Появление гибридных устройств, сочетающих в себе несколько различных способов взаимодействия, привело к потребности в адаптивных ЧМИ.

Ввиду высокой сложности модели оценки эргономики интерфейса, потребовались современный математический аппарат для её минимизации. Традиционные методы минимизации не пригодны для данной задачи ввиду специфики формализованного представления эргономических свойств ЧМИ. Разработка генетического алгоритма, способного эффективно компоновать макет в соответствии с законами эргономики позволяет решить поставленную задачу. Несмотря на широкое разнообразие возможных вариантов ГА, не все из них пригодны для эффективного решения поставленной в диссертации задачи.

Дополнительно необходимо рассмотреть актуальные технологии параллельных вычислений с применением графических ускорителей на персональных компьютерах. В качестве экспериментального подтверждения целесообразно реализовать предлагаемые методики и алгоритмы в виде приложения-верификатора и инструмента оценки уже существующих программ с графическим интерфейсом.

Объектом исследования данной диссертации является проектирование графических человеко-машинных интерфейсов для промышленных систем. Рассматривается весь процесс, начиная от формирования требований к программе и заканчивая внешним видом и средствами проектирования.

Предметом исследования выступает возможность применения генетических алгоритмов для компоновки элементов управления на экране с целью уменьшения когнитивного диссонанса оператора при взаимодействии с данным ПО.

# Аналитический обзор современных методов и средств проектирования человеко-машинных интерфейсов

## Принципы проектирования современных человеко-машинных интерфейсов (ЧМИ)

Известно [6], что развитие человеко-машинных интерфейсов происходит не только в направлении улучшения качества взаимодействия, но и в направлении поиска новых методов взаимодействия с оператором. Вместо разработки традиционных интерфейсов, разные исследования предлагают новые концепции построения взаимодействия: мультимодальность вместо унимодальности, интеллектуальные адаптивные интерфейсы вместо традиционных командных и, наконец, активные системы вместо пассивных.

В настоящее время существует обилие потребительской электроники, управляемой с помощью компьютера [7]. Поскольку функции большинства товаров потребительской электроники зачастую эквивалентны, решающую роль в выборе покупателем того или иного товара решает пользовательский интерфейс. В частности, большинство представленных на рынке мобильных телефонов выполняют одни и те же функции, разницу между ними составляет лишь способ, т.е. интерфейс, с помощью которого пользователь может к этим функциям обратиться [8].

В промышленной электронике функциональное отличие приборов пока велико, однако, учитывая тот факт, что в своём большинстве заказчик оценивает приобретаемое изделие с потребительской точки зрения, неудобный пользовательский интерфейс способен испортить первое впечатление. Именно поэтому, к интерфейсу промышленного оборудования стоит применять критерии, схожие с оценкой потребительской электроники [1].

Кроме того, эффективным способом разработки интерфейсов потребительской электроники считается использование технологии «Обновление по воздуху (OTA)» [9]. Многие из современных устройств оснащены средствами телеметрии, с помощью которых разработчик удалённо, уже после выхода устройства на рынок, может оценить удовлетворённость пользователей эргономической составляющей изделия и, при необходимости, обновить ПО дистанционно, не обременяя пользователя походом в сервисный центр [10]. В промышленном оборудовании, такая методика не применима. Поскольку промышленные приборы зачастую выполняют ответственный процесс, доступ к ним через Интернет практически всегда отсутствует, а возврат изделия в сервисный центр для обновления ПО часто влечёт за собой расходы, несоизмеримые с убытками, вызванными неудобством операторов. Поэтому интерфейс таких приборов должен сразу, начиная с первой версии, отвечать высоким эргономическим требованиям.

На данный момент существует довольно много подходов к проектированию человеко-машинного взаимодействия, однако практически каждый из них либо относится к определённому этапу проектирования, что не даёт чёткого руководства по качественному проектированию от начала и до конца, либо описывает конкретную предметную область, что лишает даный подход универсальности. Стоит также отметить, что многие из этих методик создавались во время первой волны роста популярности ПК в массах, поэтому они оценивают пользователя с точки зрения неспециалиста, т.е. не допуская, что пользователь уже может обладать определёнными навыками при работе с компьютерными интерфейсами. В дополнение к этому, на тот момент из всех доступных на сегодняшний день аппаратных средств взаимодействия были доступны только клавиатура и мышь. Это и другие несоответствия современным тенденциям ведёт к заключению о том, что большинство методик лишены адаптивности.

Особое внимание при проектировании ЧМИ промышленных систем необходимо уделять вопросам безопасности. Если в потребительской электронике основное внимание уделяется вопросам привлекательности внешнего вила и соответствия тенденциям на рынке, то в промышленных системах необходимо, в первую очередь обеспечить безопасность при выполнении производственного процесса, контролируемого прибором, и лишь затем привлекательный внешний вид.

Обобщая всё вышеперечисленное, проектирование ЧМИ промышленных систем в настоящий момент испытывает следующие проблемы:

* несогласованность этапов проектирования;
* отсутствие формализованной оценки качества;
* неопределённое время разработки;
* отсутствие адаптивной составляющей.

## Особенности существующих методов проектирования

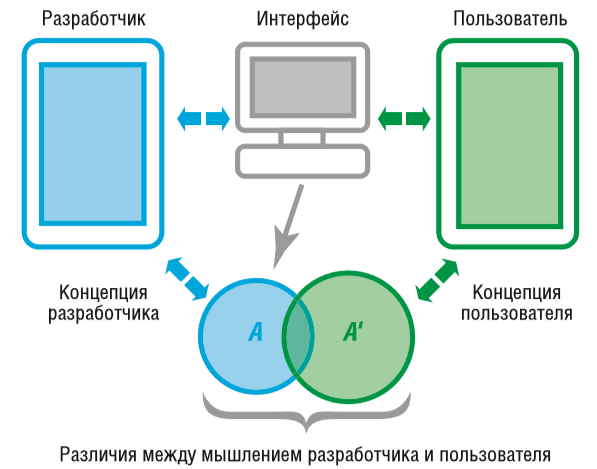


Рис.1.1. Диаграмма взаимодействия пользователя и разработчика с интерфейсом

Человеко-машинный интерфейс аналогичен взаимодействию между двумя людьми. Интерфейс играет роль сообщения, которое передаёт инженер-разработчик пользователю. Каждый из них имеет свою собственную модель восприятия, сложившуюся на основе опыта взаимодействия с окружающей средой, а потому у каждого есть своё собственное представление о том, каким рабочий интерфейс должен быть. Создавая ЧМИ, разработчик подразумевает определённые действия, с помощью которых можно добиться нужных результатов. Пользователь также знает, что конкретные действия должны иметь определённый исход, но ожидаемый им результат может не совпадать с тем, что заложил исполнитель. Таким образом возникает конфликт между ожиданиями пользователя и реальным результатом [11]. На Рис.1.1 представлена диаграмма взаимодействия пользователя и разработчика с интерфейсом. Буквой A обозначен результат, заложенный в интерфейс разработчиком (в соответствии с его концептуальной моделью), а буквой А’– результат, который ожидает увидеть пользователь. Если А и А’ совпадают (или максимально пересекаются), пользователь и разработчик одинаково видят решение, поэтому пользователь адекватно воспримет интерфейс. Если А и А’ не совпадают, разработчик не учёл концепцию пользователя и данный интерфейс не будет воспринят адекватно. Во втором случае интерфейс не будет интуитивно понятен оператору и ему придётся предпринять дополнительные усилия для понимания и запоминания, а возможно, переучиваться для работы с данным интерфейсом. Для минимизации последствий подобного конфликта интересов и создания прозрачного для пользователя дизайна HMI необходимо придерживаться рекомендаций человекоориентированного проектирования:

* учитывать нужды и ожидания конечного пользователя в соответствии с конкретными условиями;
* создавать лёгкий, простой и удовлетворяющий пользователя продукт;
* сделать технические характеристики такими, чтобы поставленные задачи решались быстро и правильно.

С 1 декабря 2013 года на территории Российской Федерации вступил в действие новый национальный стандарт ГОСТ Р ИСО 9241-210-2012 «Эргономика взаимодействия человек–система. Часть 210. Человеко-ориентированное проектирование интерактивных систем» [12]. Данный ГОСТ является переводом международного стандарта ISO 92412010:2010 «Ergonomics of human-system interaction – Part 210: Human-centered design for interactive systems” [13]. Одно из ключевых понятий, с которым работает ГОСТ, – это пригодность использования. Здесь хочется отметить, что данный термин является дословным переводом уже устоявшегося в профессиональной литературе термина «юзабилити». Юзабилити – это свойство системы, при наличии которого пользователь может использовать продукцию в определённых условиях для достижения установленных целей с необходимой результативностью, эффективностью и удовлетворённостью. Данное понятие связано с понятием эргономичности, но скорее подразумевает под собой полезность и эффективность системы и взаимодействия пользователя с ней, а не эстетическую составляющую дизайна. Основными метриками юзабилити являются эффективность, продуктивность, удовлетворённость [12]. Эффективность отображает долю пользователей, выполнивших задачу успешно. Этот показатель напрямую связан с количеством ошибок, возникающих при работе с системой. Здесь подразумеваются по большей части не ошибки системы, возникающие из-за неполадок оборудования и программного обеспечения (такие ошибки скорее будут влиять на удовлетворённость пользователя системы), а ошибки, совершаемые пользователями из-за неправильных действий. Увеличить эффективность юзабилити можно следующими способами:

* реализацией необходимых функций контроля и управления, не перегружая при этом системы ненужными встроенными опциями;
* минимизацией отношения числа ошибок к успешным действиям;
* уменьшением нагрузки на пользователя, а конкретно – на его память, внимание, зрение, воображение, слух;
* увеличением количества используемых пользователем функций и команд, то есть прозрачностью встроенных функций для пользователя.

Одним из способов увеличить эффективность, уменьшив нагрузку на пользователя, может стать использование закономерности «7±2», открытой учёным-психологом Джорджем Миллером [14]. Согласно ей кратковременная человеческая память, как правило, не может запомнить и повторить более 7±2 элементов, причём элементами могут быть слова, цифры, буквы или символы. Данную закономерность можно использовать при создании навигационного меню по системе визуализации. Так, различные подсистемы можно определить в разделы по общим признакам. Например, по системам управления — это «Энергопотребление», «Вентиляция», «Водоснабжение», «Отопление», «Канализация», «Освещение», «Система доступа». Здесь также работает закон Хика: чем меньше элементов меню, тем меньше времени занимает выбор одного из них [15]. Продуктивность отображает количество ресурсов, которые пользователь затрачивает для достижения поставленных целей. К этим ресурсам относят:

* время, необходимое для выполнения пользователем определённой задачи;
* время, необходимое на предварительное обучение перед использованием мнемосхемы;
* частота использования справки или документации;
* количество повторных и ошибочных действий.

Уменьшить время, необходимое для выполнения пользователем определённой задачи, можно с помощью закона Фиттса, опубликованного в 1954 году Полом Фиттсом [16]. Рассмотрим ситуацию, включающую в себя объект, управляемый человеком, например, курсор мыши, и цель, например, кнопку экстренной остановки оборудования, расположенную где-то в другом месте экрана. Математически закон Фиттса можно записать следующим образом [17]:

где MT – среднее время, затрачиваемое на совершение действия, c; a – время запуска/остановки устройства, c; b – величина, зависящая от типичной скорости устройства, бит/с; A – дистанция от точки старта до центра объекта; W – ширина объекта, измеренная вдоль оси движения. Согласно формуле, чем дальше инструмент управления (курсор мыши, стилус сенсорного экрана) находится от цели и чем меньше размер цели, тем больше времени потребуется для его позиционирования. Наличие логарифмической зависимости также указывает на то, что, хотя небольшое увеличение размеров малых объектов позволяет их легче выделять, изменение размеров больших объектов уже не имеет значения. То же самое касается и дистанции до цели. Удовлетворённость определяет комфорт и приемлемость пользования продуктом, а также благополучие эмоционального состояния пользователей после выполнения задачи. Это определяется:

* полезностью и функциональностью HMI;
* удобством использования;
* оценкой того, насколько технологическое выполнение задачи соответствует требованиям по контролю и управлению HMI.

Например, на удовлетворённость пользования системой влияет наличие предыдущего опыта пользователя. В психологии известен синдром, имеющий название «Синдром утёнка» или «импринтинг» [18]. Обычно пользователи привязываются к первому изученному ими принципу работы системы и все остальные системы судят по тому, насколько они на него похожи. В этом случае использование какого-либо нового принципа может вызвать у человека резкое неприятие системы визуализации и уменьшение удобства использования. Это относится и к новым функциям, к которым пользователь не привык и поэтому они для него неудобны.

Системы автоматизации обрабатывают большое количество информации, поступающей из различных источников. Качество и количество информации влияет на принятие управленческих решений. Большое количество отображаемых данных не означает, что пользователь будет более осведомлённым. Даже наоборот, пользователь может быть введён в заблуждение. Правильный интерфейс должен подавать информацию таким способом, чтобы обеспечить выполнение нужных целей, задач и потребностей пользователя. Человеко-ориентированное проектирование базируется на следующих принципах.

1. Технология должна быть построена с учётом целей, задач и возможностей пользователя. Необходимо убедиться, что система предоставляет все необходимые данные и исключает лишнюю информацию, которая может ввести в заблуждение.
2. С организационной стороны технология должна учитывать то, как пользователь обрабатывает информацию и принимает решения. В случае возникновения некорректных ситуаций пользователи пытаются определить и понять происходящее событие. Эксперты пользуются шаблонным набором действий, хранящимся в долговременной памяти, для того чтобы быстро разобраться в ситуации. Лица, принимающие решения, должны понимать не просто происходящее событие, а разобраться в ситуации в целом.
3. Пользователи должны быть осведомлены о состоянии системы, чтобы иметь возможность контролировать её. В случае недостаточной осведомлённости пользователя о состоянии системы это может негативным образом повлиять на принятие решения. Ситуационная осведомлённость (SA – Situational Awareness) – это понимание оператором состояния системы в данный момент времени, которое основывается на персональном понимании, приобретённых знаниях и имеющейся информации. Правильность принятия решений напрямую зависит от ситуационной осведомлённости [19].

Ситуационная осведомлённость подразумевает три составляющие:

* пользователь в курсе того, что происходит;
* он понимает то, что эта информация означает;
* он предполагает, что эта информация будет значить в будущем.

Таким образом, ситуационная осведомлённость подразделяется на три уровня:

1-й уровень (1 SA) – восприятие необходимых данных. Здесь имеются в виду состояние, свойства, динамика изменения

элементов. В частности, это отображение состояния оборудования и датчиков.

2-й уровень (2 SA) – понимание текущей ситуации. Понимание основано на восприятии данных с первого уровня и интерпретации их в соответствии с установленными целями. Неопытные проектировщики могут не иметь достаточной базы знаний для разработки системы со вторым уровнем осведомлённости. Ко второму уровню могут быть отнесены данные, отображаемые на шкалах и счётчиках. Обычные текстовые значения сложны для восприятия. Человек гораздо быстрее воспринимает и интерпретирует графическую информацию. Абсолютные цифровые данные несущественны, если они не сопоставляются с предельными значениями [20].

3-й уровень (3 SA) – проекция будущего состояния. Здесь имеется в виду возможность предсказать, что произойдёт дальше. Это может быть достигнуто только за счёт хорошего понимания уровня 2 SA. К уровню 3 SA относится отображение трендов, так как с их помощью пользователь может проанализировать тенденцию изменения работы системы и таким образом экстраполировать данные на перспективу.

Существуют негативные факторы, которые подрывают ценность осведомлённости [21].

1. Туннелированное внимание – фиксация на одном неполном наборе информации в ущерб другим. Это наиболее распространённая ошибка.
2. Ловушка рабочей памяти – надежда на ограниченную кратковременную память.
3. Рабочая нагрузка, беспокойство, усталость и другие стрессовые ситуации. Всё это уменьшает возможность человека адекватно обрабатывать информацию.
4. Перегрузка информацией. Слишком большое количество информации уменьшает осведомлённость. Например, текстовые данные гораздо хуже воспринимаются, чем информация, представленная графически.
5. Смещение акцентов: красный цвет, движущиеся объекты, мигающий свет могут отвлечь человека от важной информации. Поэтому данные средства должны быть использованы в случаях, если пользователя надо уведомить о чрезвычайных ситуациях.
6. Сложность системы. Слишком большое количество встроенных функций усложняет восприятие системы.
7. Неправильная модель системы ведёт к ложной трактовке событий.

Большой вклад в развитие человеко-машинного взаимодействия внесла модель GOMS (Goals, Operators, Methods, Selections rules) представленная в 1983. GOMS до сих пор является наиболее широко известным методом проектирования и оценки удобства человеко-машинных интерфейсов, поскольку он выдаёт численные и качественные характеристики эргономичности интерфейса с точки зрения людей.

Модель GOMS состоит из методов достижения определённых целей. В основе методов лежат операторы. Операторами называются определённые шаги, которые выполняет пользователь. Каждому оператору назначается нормативное время выполнения. Если цель может быть достигнута несколькими способами, то применяются правила выбора, которые определяют наиболее подходящий вариант.

* Цели – символьные структуры, определяющие конечное состояние системы и перечень возможных путей достижения.
* Операторы – элементарные, двигательные, когнитивные действия и действия восприятия, выполнение которых необходимо для изменения пользовательского восприятия или состояния системы.
* Методы – описания процедур достижения цели.
* Правила выбора – правила по которым определяется наиболее подходящий метод достижения цели.

Существует несколько разновидностей модели GOMS, каждая из которых делает акцент на отдельном аспекте ЧМИ. Тем не менее, основные концепции всех разновидностей совпадают. Допускается определённая гибкость, например, оператор в одном методе может являться целью в другом. Уровень детализации выбирают таким образом, чтобы отследить требуемый уровень объектов.

Существует четыре разновидности модели GOMS: Keystroke-Level Model, CMN-GOMS, NGOMSL и CPM-GOMS. Каждая модель подразумевает различные уровни сложности, и разнятся по видам деятельности.

Keystroke-Level Model (KLM, модель уровня клавишного ввода) – самая первая и наиболее простая техника GOMS. Теоретическое и практическое время выполнения задачи оценивается путём составления последовательности операторов и суммирования нормативного времени каждого их них. В случае KLM модели, необходимо явно указывать метод выполнения задачи. Более того, методы должны быть представлены последовательностями клавишных операций. KLM предполагает четыре типа двигательных операций:

* K – ввод текста или нажатие определённых клавиш;
* P – наведение курсора на объект;
* H – перемещение руки на клавиатуру;
* D – проведение отрезка на сетке.

Дополнительно вводятся ментальный оператор M, олицетворяющий время, необходимое пользователю на мыслительную подготовку к совершению действия, и оператор реакции системы R, отображающий ожидание пользователем реакции системы. Время исполнения – это сумма отрезков времени, затраченных на каждый отдельный вид операции:

Каждый из перечисленных операторов обладает нормативным временем исполнения, определённым либо числовой константой, либо функцией с параметром.

Традиционная модель KLM разрабатывалась во времена клавиатурных интерфейсов. Поэтому с появлением сенсорных систем ввода появилась потребность в новой модели оценки взаимодействия. Это привело к разработке Touch Level Model (TLM, модель сенсорного уровня). Эндрю Райс и Джонатан Лартигью предложили эту модель для оценки пользовательского взаимодействия на устройствах с сенсорным экраном. Исследования показали высокую точность предсказания. Целью разработки было предоставить инструмент для количественной оценки удобства сенсорных интерфейсов. Список операций включает в себя:

* отвлекающее воздействие (X) – мультипликативный фактор, применяемый к другим операторам для учёта отвлекающих воздействий окружающей среды;
* жест (G) – жесты описываются как особые комбинации движений пальцев по экрану;
* щипок (P) – один из основных двупальцевых жестов;
* увеличение (Z) – оператор, противоположный P;
* Начальное воздействие (I). В KLM предполагалось, что устройство всё время готово к взаимодействию с пользователем. Однако в сенсорных устройствах перед воздействием требуется подготовка (нажатие кнопки «Домой» или ввод пароля);
* касание (T) – касание экрана в отдельной области для совершения действия или внесения изменения;
* пролистывание (S) – горизонтальное или вертикальное пролистывание, наподобие пролистывания книги;
* наклон (L(d)) – производится при взаимодействии с устройствами, снабжёнными акселерометром;
* вращение (O(d)) – жест, при котором два и более пальцев помещаются на экран, а затем проворачиваются вокруг центральной точки;
* перетаскивание (D) – жест наподобие пролистывания, включающий в себя касание определённой точки на экране, а затем перемещение пальца по экрану в определённом направлении.

CMN-GOMS – оригинальная модель, построена на базе KLM и добавляет в неё подзадачи и правила выбора методов. Эта модель позволяет предсказать последовательность операторов и время их исполнения. Модель CMN-GOMS может быть реализована в виде программы, что даёт возможность не только анализировать, но и выполнять её. CMN-GOMS использовалась при проектировании эргономических аспектов текстовых процессоров и CAD систем. CMN метод позволяет количественно предсказать порядок операторов и время их выполнения, а также заостряет внимание на методах достижения целей. Пример CMN-GOMS правки рукописи:

* ЦЕЛЬ: РЕДАКТИРОВАТЬ-РУКОПИСЬ
  + ЦЕЛЬ: РЕДАКТИРОВАТЬ-ГЛАВУ
    - ЦЕЛЬ: ПОЛУЧИТЬ ГЛАВУ
      * ЦЕЛЬ: ПЕРЕЛИСТНУТЬ-СТРАНИЦУ
      * ЦЕЛЬ: ПОЛУИТЬ-СОДЕРЖИМОЕ
    - ЦЕЛЬ: ВНЕСТИ-ИЗМЕНЕНИЯ
      * ЦЕЛЬ: ИСПРАВИТЬ-ТЕКСТ
        + Выбор: ЦЕЛЬ: ПЕРЕМЕСТИТЬ-ТЕКСТ
        + ЦЕЛЬ: УДАЛИТЬ-ФРАЗУ
        + ЦЕЛЬ: ВСТАВИТЬ СЛОВО
      * ПРОВЕРИТЬ-ПРАВКУ

NGOMSL – структурированная нотация на естественном языке для представления GOMS моделей, а также процедур для их составления. Эта программная форма предоставляет предсказания последовательностей операторов, времени исполнения и времени обучения методам. Модель NGOMSL составляется сверху-вниз по целям пользователя, формируя при этом методы, которые впоследствии состоят из примитивных операторов, в основном клавишных. Модель однозначно представляет структуру целей наподобие CMN-GOMS. Пример:

* МЕТОД для ЦЕЛИ: ПЕРЕМЕСТИТЬ ТЕКСТ
  + ШАГ 1: ДОСТИЧЬ ЦЕЛИ: ВЫРЕЗАТЬ ТЕКСТ
  + ШАГ 2: ДОСТИЧЬ ЦЕЛИ: ВСТАВИТЬ ТЕКСТ
  + ШАГ3: ВОЗВРАТ С ДОСТИГНУТОЙ ЦЕЛЬЮ
* МЕТОД для ЦЕЛИ: ВЫРЕЗАТЬ ТЕКСТ
  + ШАГ 1: ДОСТИЧЬ ЦЕЛИ: ВЫДЕЛИТЬ ТЕКСТ
  + ШАГ 2: ЗАПОМНИТЬ КОМАНДУ «ВЫРЕЗАТЬ» И ДОСТИЧЬ ЦЕЛИ: ВЫПОЛНИТЬ КОМАНДУ
  + ВОЗВРАТ С ДОСТИГНУТОЙ ЦЕЛЬЮ
* …

Существуют четыре различных типа GOMS [2]. CMN-GOMS, KLN и NGOMSL предполагают, что все операторы выполняются последовательно. CPM-GOMS, будучи четвёртым методом, рассматривает операторы с точки зрения Человеческой Модели Обработки (MHP), которая подразумевает, что когнитивный, двигательный и оператор восприятия могут выполняться человеком параллельно. Главной особенностью CPM-GOMS является возможность предсказать поведение опытного пользователя, благодаря моделированию пересекающихся действий. Рассмотрим модель на примере операции копирования-вставки:

* ЦЕЛЬ КОПИРОВАТЬ-ВСТАВИТЬ-ТЕКСТ
  + ЦЕЛЬ КОПИРОВАТЬ-ТЕКСТ
    - ЦЕЛЬ ВЫДЕЛИТЬ-ТЕКСТ
      * Оператор ПЕРЕДВИНУТЬ-КУРСОР-НА-НАЧЛО
      * Оператор ЩЁЛКНУТЬ-МЫШЬЮ
      * Оператор ПЕРЕДВИНУТЬ-КУРСОР-НА-КОНЕЦ
      * Оператор SHIFT-КЛИКНУТЬ-МЫШЬЮ
      * Оператор ПРОВЕРИТЬ-ВЫДЕЛЕНИЕ
    - ЦЕЛЬ ВЫПОЛНИТЬ-КОМАНДУ-КОПИРОВАТЬ
      * Выбор
        + ЦЕЛЬ ИСПОЛЬЗОВАТЬ-МЫШЬ

Оператор ПЕРЕМЕСТИТЬ-КУРСОР-К-МЕНЮ-ПРАВКА

Оператор ЩЁЛКНУТЬ МЫШЬЮ

Оператор ПЕРЕМЕСТИТЬ-КУРСОР-К-ПУНКТУ-КОПИРОВАТЬ

Оператор ПРОВЕРИТЬ-ВЫДЕЛЕНИЕ

Оператор ОТПУСТИТЬ-КНОПКУ-МЫШИ

* + - * + ЦЕЛЬ ИСПОЛЬЗОВАТЬ-КЛАВИАТУРУ

Оператор НАЖАТЬ-КЛАВИШУ-CTRL

Оператор НАЖАТЬ-КЛАВИШУ-C

Оператор ОТПУСТИТЬ-КЛАВИШИ

* + - ЦЕЛЬ ВСТАВИТЬ-ТЕКСТ […]
* Правило выбора ЦЕЛИ ВЫПОЛНИТЬ-КОМАНДУ-КОПИРОВАТЬ
* Если РУКИ-НА-КЛАВИАТУРЕ тогда
  + Выбор ЦЕЛЬ ИСПОЛЬЗОВАТЬ-КЛАВИАТУРУ
* Иначе

Выбор ЦЕЛЬ ИСПОЛЬЗОВАТЬ-МЫШЬ

«Теория деятельности» представляет собой более описательную мета-теорию, чем теорию предсказаний. В ней рассматривается не только действие, совершаемое отдельно взятым пользователем, но и поведение всей системы в целом. В систему входят окружение, история пользователя, культуры, роль объекта взаимодействия, мотивации и сложность осуществления действия в реальной жизни. Одной из сильных сторон данной методики является решение проблемы связи индивидуального субъекта и общества. Объектом анализа в «Теории деятельности» выступает концепция объектно-ориентированной коллективной, культурно образованной системы.

Цель «Теории деятельности» – понимание ментальных способностей отдельного индивидуума. Однако, она отбрасывает изолированных индивидуумов, анализируя культурные и технические аспекты действий человека. «Теория деятельности» основывается на шести принципах:

* объекты – реальность описывается множеством взаимосвязанных объектов;
* субъекты – актёры, выполняющие исследуемые действия;
* общество – социальный контекст, в котором существуют актёры;
* инструментарий – артефакты или концепты, используемые актёрами;
* разделение труда – социальная страта, иерархическая структура действия;
* правила – конвенции, руководства и правила, регулирующие использование системы.

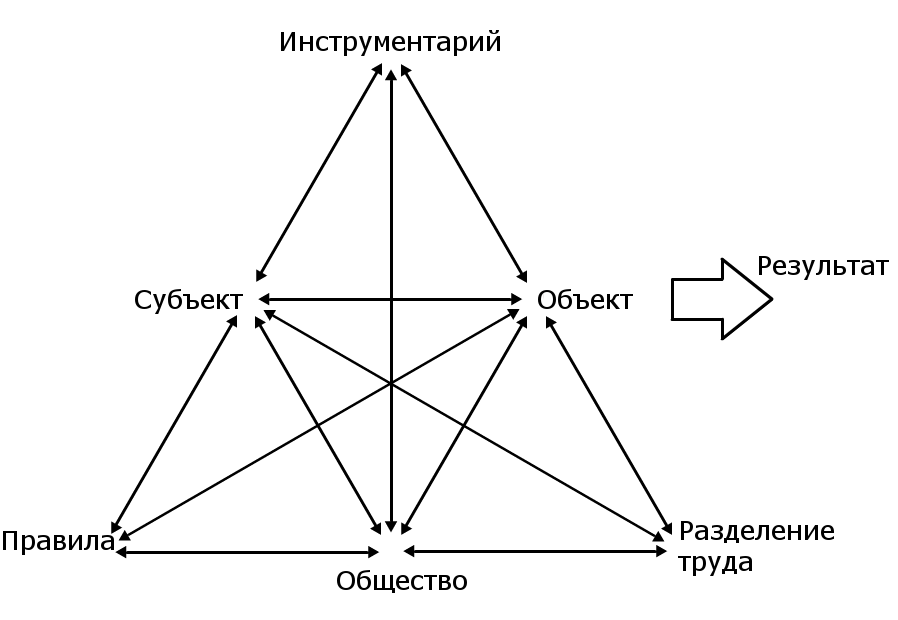


Рис.1.2. Компоненты «Теории деятельности» и их взаимосвязи

Рост числа персональных компьютеров изменил направление компьютерных разработок с автоматизации существующих задач на мейнфреймах на то, как пользователи хотят взаимодействовать с объектами с помощью компьютера. В поисках теоретических и методологических перспектив, способствующих решению вопросов гибкости и взаимодействия человека с машиной, было принято решения о применении уже существующей «Теории деятельности» к зарождающейся науке ПРЧМИ.

«Теория деятельности» начинается с описания действия. Действие рассматривается как система взаимодействия человека, т.е. субъекта, совершающего влияние на объект, с целью получения результата. Для этого, субъект использует инструментарий, который может быть, как внешним (например, топор, компьютер и т.д.) или внутренним (например, план действий). В качестве примера можно привести работу автоматического call-центра. В данном примере участвуют несколько субъектов, каждый их которых, в конечном итоге, преследует одну или несколько целей (повышение эффективности поставок, карьерный рост или получение контроля над важным источником власти). Примером такого действия является телефонный оператор (субъект), исправляющий учётные данные клиента (тоже субъекта) таким образом, чтобы данные счёта были верными (результат), при этом используя графический клиент доступа к базе данных (инструментарий).

Действие представляется в виде трёхуровневой модели.

* инструментарий;
* правила;
* разделение труда.

## Существующие средства проектирования

На сегодняшний день основным средством проектирования выступает персональный компьютер со специализированным ПО.

Microsoft Visual Studio и Blend для Visual Studio предоставляют средства визуального проектирования нагруженных ЧМИ с широкими мультимедийными возможностями. Оба инструмента имеют много общего (например, редактор XAML), однако Blend для Visual Studio содержит дополнительные инструменты для создания анимированных и поведенческих ЧМИ. Blend позволяет разработчикам, знакомым с внешним видом среды программирования Visual Studio без лишних сложностей заниматься разработкой графической составляющей своей программы. Примечательной особенностью является назначенная по умолчанию тёмная тема интерфейса. По заявлению разработчиков, это позволяет лучше сконцентрироваться на создаваемом объекте, а не на окружающем его инструментарии.

Blend, также как и Visual Studio, позволяет выполнять отладку программы. Это удобно в тех случаях, когда изменения в пользовательском интерфейсе затрагивают логику работы программы, что в целом неправильно, но, на практике, неизбежно. Оба рассматриваемых средства поддерживают перезагрузку открытых документов, что необходимо при параллельном использовании одного файла в двух программах. Среди прочего Blend предоставляет возможности визуального редактирования анимаций. Это позволяет выстраивать плавные переходы в ЧМИ по раскадровкам, не прибегая к необходимости писать программный код. Дополнительно данная среда позволяет преобразовать любой элемент управления в векторный образ, что обеспечивает возможность произвольной настройки любого элемента управления. Механизм поведений позволяет «оживить» статический дизайн макета без потребности в традиционном программировании. Кроме того, поведения позволяют логически скомпоновать и повторно использовать общепринятые концепции в работе элементов управления. В частности, подсветка кнопки при наведении курсора может быть настроена один раз, а затем повторно применяться к другим элементам, унаследованным от того же родителя, например пункт меню, элемент списка и т.д.

Наиболее актуальной с точки зрения данной диссертации возможностью Blend является редактирование шаблонов представления любого элемента управления. Такая парадигма носит название «Lookless controls». С помощью данной технологии можно настроить отображение ЧМИ именно так, как этого требует предметная область.

В качестве технологии, ускоряющей процесс разработки ЧМИ, в состав Blend входит инструментарий прототипирования. Как известно, большую часть времени разработки занимает отладка. Рассмотрим случай, когда разрабатывается индикатор состояния определённого прибора. Чтобы проверить работу данного индикатора (иными словами, оценить как он отображает различные состояния системы: от удовлетворительного до критического), программисту приходится выстраивать дополнительный уровень абстракции для имитатора. В случае с Blend специальный имитатор позволяет, даже не запуская целевую программу, сымитировать различные состояния системы.

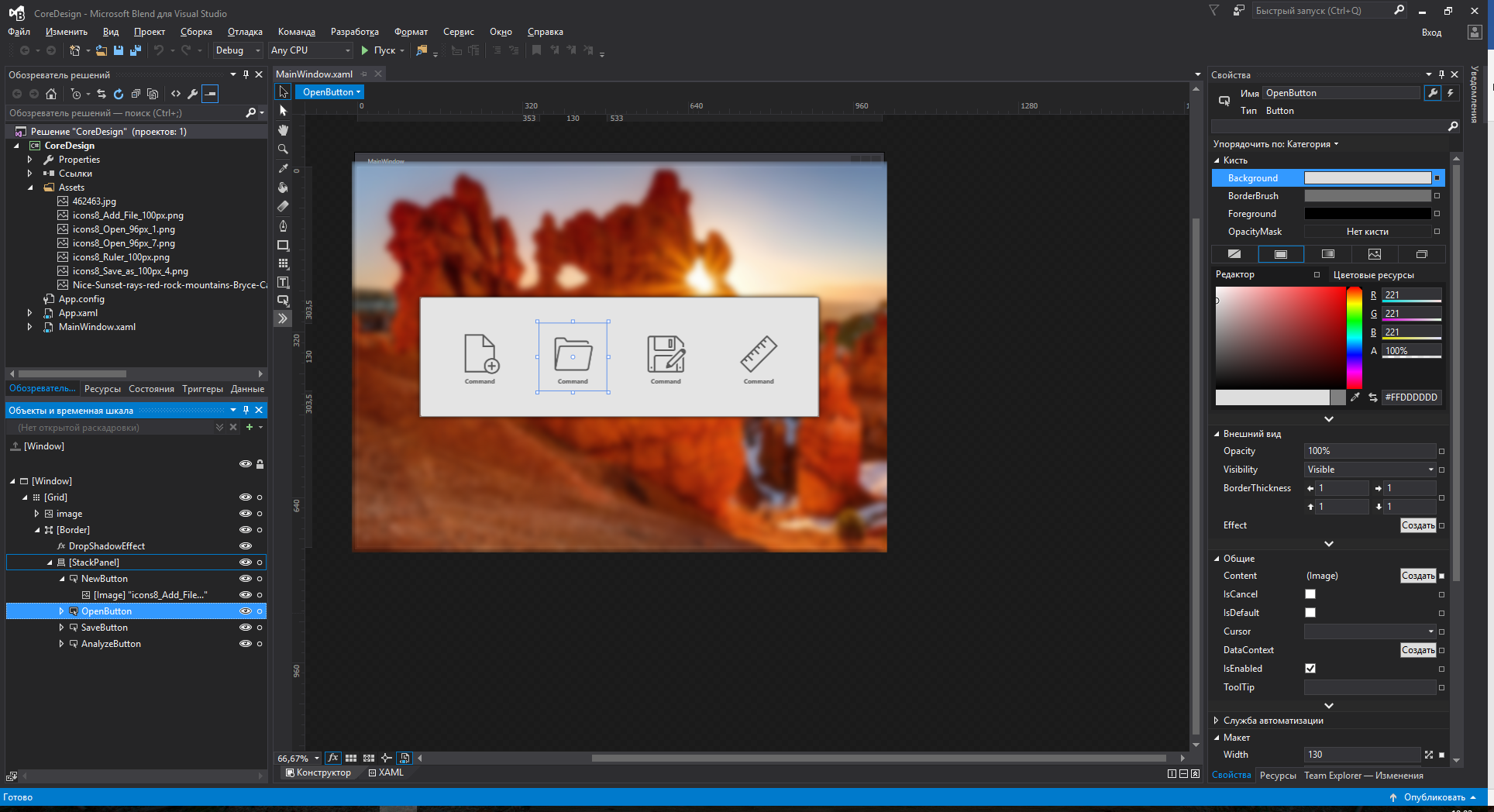


Рис.1.3. Визуальный подход к разработке ЧМИ

Также актуальным на сегодняшний день является визуальный редактор Glade, входящий в состав графической среды Gtk+. Изначально, графические интерфейсы в Gtk+ создавались в коде. Это означает, что этап визуального проектирования, как таковой, системой не предусматривался, а выполнялся программистом с помощью подручных средств. В конечном итоге интерфейс кодировался с помощью команд на языке C (впоследствии был также предложен язык Vala, но существенно этот шаг ситуацию не улучшил). В результате долгого развития среды Gtk+ в определённый момент была сформирована технология описания макета с помощью XML документов. И уже для автоматической генерации XML разметки был создан визуальный редактор Glade.

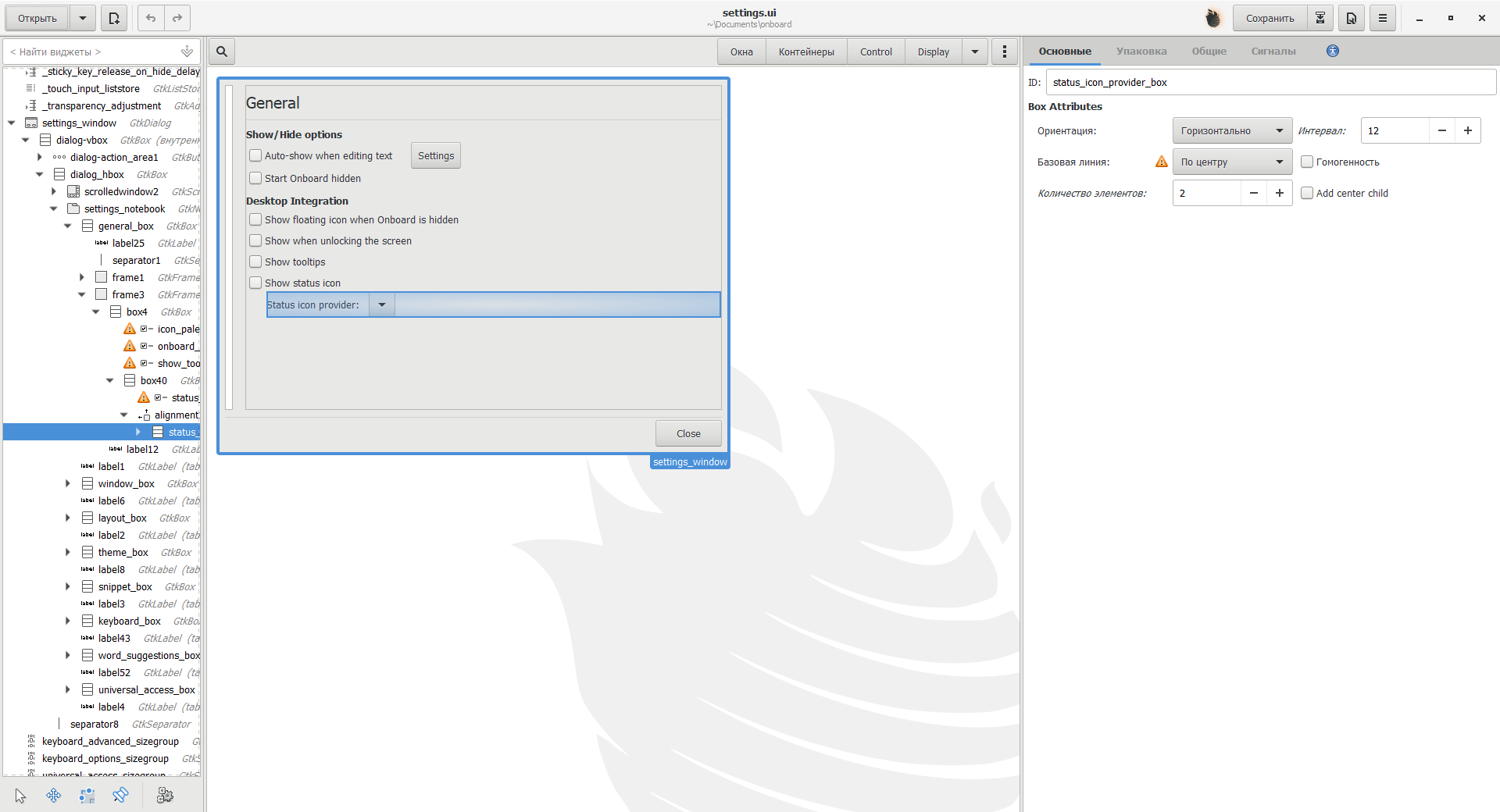


Рис.1.4. Разработка ЧМИ с помощью кодирования

Стоит отметить, что кодирование пользовательского интерфейса с помощью языка программирования не являлось чем-то особенным на тот момент. Учитывая то, что графические интерфейсы служили концептуальным развитием человеко-машинного взаимодействия в целом, логично предположить, что макет, как и любой другой объект управления, может контролироваться программным кодом. Такой подход до сих пор используется во многих других средствах проектирования, например «Windows Forms», «Swing», «Qt». Такой подход априори создаёт барьер для дизайнеров. В итоге человек, желающий проектировать, должен обладать навыками программирования.

## Анализ преимуществ и недостатков современных ЧМИ

Модель GOMS имеет свои сильные и слабые стороны. В то время как GOMS не является самым достоверным методом оценки эргономики ЧМИ, он отражает все процедурные знания. С помощью GOMS можно без особых затруднений рассчитать пользовательское взаимодействие и выполнить оценку скорости довольно легко. Это становится возможным при наличии предварительно установленных экспериментальным путём эталонных значений времени, которые тратит пользователь на индивидуальные операции. С повышением точности этих значений увеличивается точность расчётов.

Из недостатков следует учесть тот факт, что модель GOMS применима только к опытным пользователям. Она не применима к начинающим пользователям ввиду того, что она не учитывает ошибки пользователя при взаимодействии с системой. Также модель не учитывает выработку навыков работы с системой у пользователя, а также долгое время неиспользования системы. Ментальная нагрузка также не учитывается в данной модели, тем самым оставляя этот параметр непредсказуемым. Тоже самое касается фактора усталости оператора. GOMS оценивает эргономику выполнения определённых задач в системе, а не функциональность в целом. Пользовательские особенности, привычки, или физические ограничения (для людей с ограниченными возможностями) не учитываются ни в одной разновидности GOMS. Все пользователи учитываются, как одна модель.

Большинство научных теорий о пользовательских интерфейсах упускают проблемы, связанные с эмпирическим представлением проектов. Во-первых, многие сложные интерфейсы представляют в роли пользователя самого разработчика, или некоего среднего пользователя, без учёта его специальности, рабочего окружения, рода деятельности и т.д. Во-вторых, при разработке часто делается упор на нового пользователя, ещё не знакомого с системой, в то время как в повседневной жизни найдутся опытные пользователи, чьё взаимодействие не рассматривается. В-третьих, детальная проработка задач на примере идеализированных условий упускает сложность реальных примеров. Кроме того, традиционные модели всегда рассматривали только модель «Один пользователь – один компьютер».

## Постановка задачи диссертационного исследования

На основании вышесказанного и учитывая недостатки существующих методов и средств проектирования ЧМИ, в ходе диссертационного исследования необходимо:

* разработать методику проектирования человеко-машинных интерфейсов промышленных систем;
* разработать формализованное представление задачи оценки эргономичности ЧМИ;
* разработать алгоритм реализации навигационной парадигмы ЧМИ;
* разработать алгоритм формирования макета операторской панели;
* разработать программную реализацию предложенных алгоритмов и методов.

Выводы по главе I

1. Определена проблемная ситуация, заключающаяся в том, что актуальные на сегодняшний день принципы проектирования не обеспечивают оптимального сочетания следующих критериев:
   * визуальная согласованность элементов управления;
   * предсказуемое время разработки;
   * достоверность оценки качества;
   * адаптивность к различным способам ввода/вывода.
2. Рассмотрены принципы проектирования современных человеко-машинных интерфейсов (ЧМИ).
3. Исследованы существующие методы проектирования.
4. Произведён обзор существующих средств проектирования.
5. Проведён анализ преимуществ и недостатков современных ЧМИ.

# Формализация задачи оптимизации эргономики ЧМИ

## Анализ эргономических требований

Для решения задачи оптимизации эргономики ЧМИ, необходимо сначала определить из каких виды эргономики состоит человеко-машинное взаимодействие.

Эргономика промышленных систем, где применяется дополнительное оборудование, строится вокруг естественных размеров и возможностей человеческого тела. Директива [22] гласит, что принцип эргономики допускает вариативность размеров оператора, его силы и выносливости. Существуют европейские стандарты [23] и [24] руководствующие проектирование в соответствии с указанной выше директивой. Европейский стандарт [25] устанавливает целью проектирования устройств, в контексте промышленной системы, согласованность с человеческими возможностями, ограничениями и потребностями. Это требует анализа производственных задач на предмет влияния возможных ограничений проектирования окружающей среды (такие как шум, вибрация) на здоровье, безопасность и самочувствие операторов. Интерфейс должен быть спроектирован с учётом вариативности следующих параметров оператора:

* габариты тела;
* осанка;
* движения тела;
* физическая сила;
* мыслительные способности [25].

### Ограничения мыслительной деятельности

Учёт размеров тела требуется во избежание попадания пользователя в опасные части устройств прибора. В частности, если предполагается, что оператору запрещается близко подходить к оборудованию, то элементы пользовательского интерфейса должны быть выполнены в крупном масштабе для облегчения различимости важной информации операторами с разным уровнем зрения с большого расстояния.

Документы [25] и [26] описывают эргономические требования в отношении мыслительной деятельности при использовании промышленного оборудования для интерфейсов задач и взаимодействия соответственно. Среди рассматриваемых способностей присутствуют внимание, восприятие (включая память и умение рассуждать) и скорость реакции. Далее будут рассмотрены следующие свойства:

* внимание;
* ощущения;
* память;
* способность рассуждать;
* рефлексы.

Человеческое внимание подразделяется на выборочное и разделяемое, в зависимости от рассматриваемой задачи и сопутствующих факторов. Способность человека разделять своё внимание между несколькими параллельными процессами на практике довольно ограничена и приводит к ошибкам. Системы управления и контроля, в которых человеку приходится обрабатывать потоки информации, должны либо отображать потоки максимально близко друг к другу, либо задействовать различные органы чувств (зрение, слух, осязание, обоняние и т.д.).

Получив информацию через органы чувств, человек классифицирует и группирует её в соответствии с определёнными правилами. Эти правила относятся к Гештальтпсихологии. В частности, предметы воспринимаются как единая группа, если они движутся в одном направлении (принцип смежности), имеют схожую форму (принцип схожести) или образуют простой, правильный и упорядоченный рисунок (принцип целостности). При проектировании ЧМИ, в частности панелей ручного управления, следует руководствоваться этими принципами.

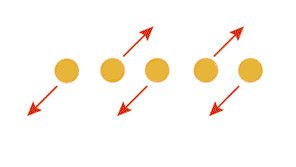


Рис.2.1. Закон общей тенденции

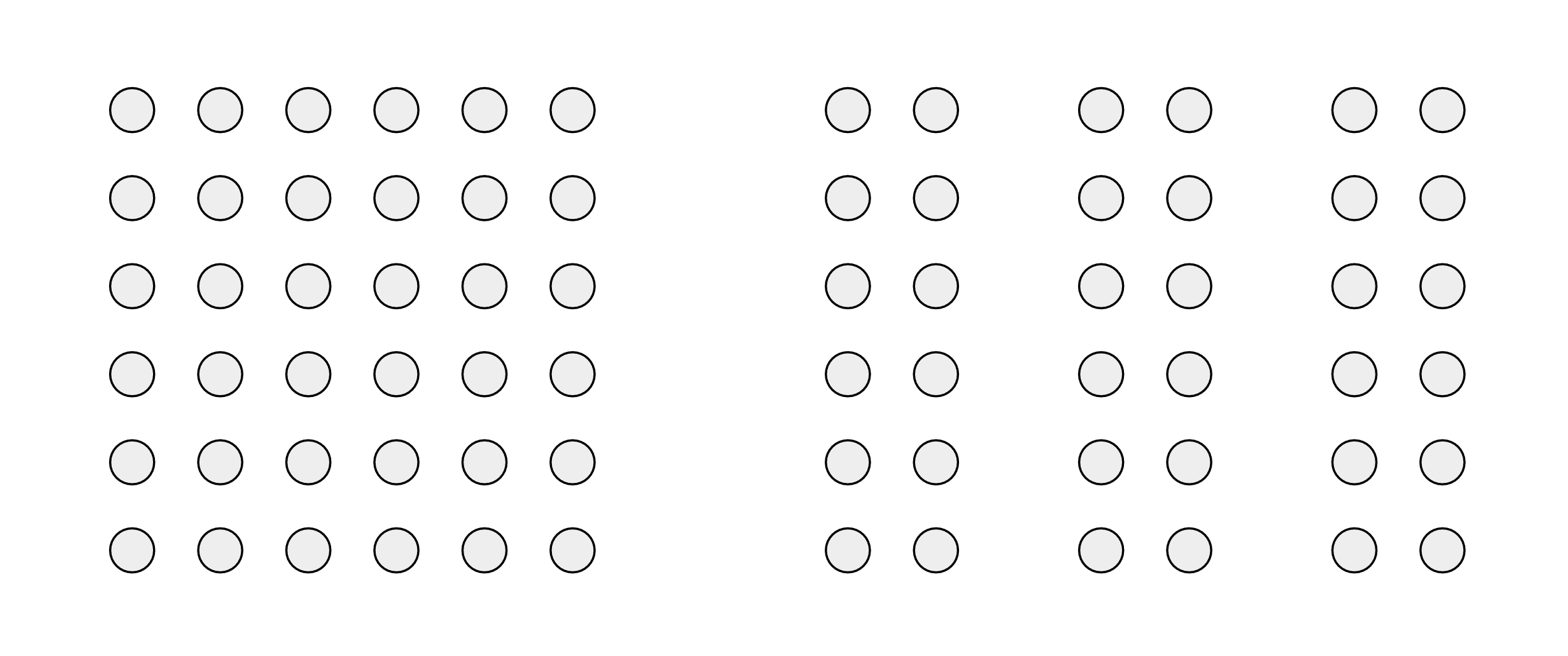


Рис.2.2. Закон близкого расположения

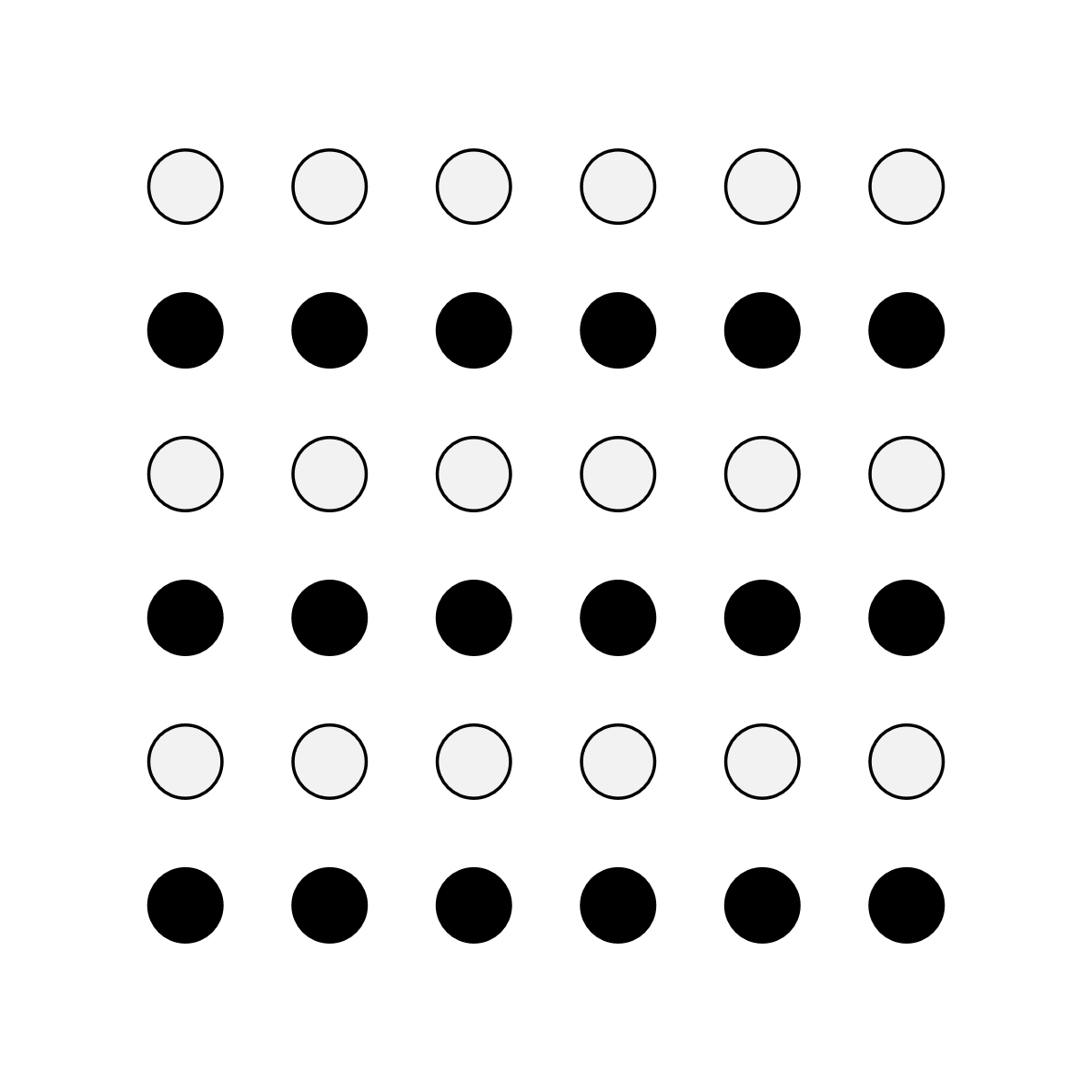


Рис.2.3. Закон схожести

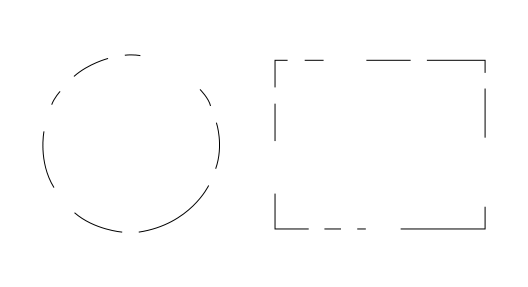


Рис.2.4. Закон замыкания

### Анализ модели памяти человека

В соответствии с современным представлением, человеческая память состоит из трёх частей. Однако эти части не отделены друг от друга, а работают вместе. Раздражитель, воспринятый органом чувств, сначала попадает в сенсорную память. Информация в сенсорной памяти хранится от долей секунды до 3,5 секунд в зависимости от силы раздражителя. Сенсорная память хранит информацию для обеспечения возможности распознавания последовательностей сигналов. Вся неважная информация отсеивается на данном этапе. Вся важная информация передаётся далее в «быструю» (рабочую) часть. Данная часть может удерживать довольно ограниченный объём информации в течение нескольких секунд. Важная информация, в свою очередь, снова передаётся в следующую часть – на этот раз в долгосрочную память. Объём долгосрочной памяти безграничен, однако могут возникать трудности с извлечением информации из неё. Извлечение происходит намного быстрее, когда необходимо распознать, а не вспомнить что-либо. Зачастую пользователям проще найти нужную клавишу, посмотрев на панель, чем назвать её не глядя. В соответствии с данными особенностями человеческой памяти, вся информация, с которой нужно оперировать пользователю при работе с оборудованием, должна быть представлена в виде, учитывающем важность этой информации обеспечения производственного процесса. Также необходимо правильно рассчитывать длительность отображаемых сообщений, чтобы пользователь успел правильно среагировать на них. Немаловажным является визуальное сходство отображаемой информации реальным частям оборудования.

Возможность пользователя классифицировать воспринимаемой информации сильно зависит от их жизненного опыта. Если у оператора отсутствует понимание того, как работает агрегат, то он вынужден самостоятельно добыть эту информацию и сформировать умозрительное заключение. Такая форма мыслительного процесса называется «снизу-вверх». Она считается затратной по времени и требует усилий от конечного пользователя. С противоположной стороны форма «сверху-вниз» имеет место, когда пользователь уже обладает представлением об устройстве прибора или вариантах его функционирования. В таком случае теоретическое знание даёт возможность пользователю быстро вникнуть в работу прибора. Однако представление об устройстве прибора может быть причиной ошибок оператора. Так происходит в случаях, когда реализация прибора имеет различия с общей концепцией. Также вероятность ошибки из-за неправильного представления о приборе увеличивается в условиях спешки. Хорошей практикой при проектировании оборудования считается следование обоим принципам одновременно.

В соответствии со стандартом [26] неосознанный рефлекс занимает 0,04 с. Осознанные действия, включающие мыслительную деятельность, занимают как минимум 0,15 с. Если раздражитель внезапен, то время реакции увеличивается как минимум до 0,5 с.

Стандарт [26] также упоминает переутомление как фактор, снижающий производительность пользователя системы. Другим важным аспектом при разработке ЧМИ является предотвращение негативных последствий переутомления, будь то последствия для здоровья или повышенный риск аварии на производстве.

Несмотря на большой перечень определений нагрузки, существует общая международная договорённость касательно умственной нагрузки, обозначенная в [27]. Взаимосвязь между умственной нагрузкой и переутомлением описывается также в [27]. Принципы эргономики, препятствующие возникновению последствий переутомления, таких как умственная усталость, монотонность, снижение бдительности, а также снижение работоспособности представлены в [28]. Целью данных документов является создание «оптимальных рабочих условий с соблюдением требований безопасности и охраны здоровья, самочувствия, сохранения производительности и предотвращения как перегрузки, так и недостаточной нагрузки. Эти принципы могут быть отнесены к ЧМИ по причинам, описанным ниже.

Переутомление возникает в результате длительной интенсивной умственной нагрузки, в течение долгого времени. Если слишком много информации сообщается пользователю за короткое время (фактор мыслительного стресса), то ограниченная ёмкость рабочей памяти оказывается исчерпанной (индивидуальная характеристика каждого человека), что привозит к переутомлению, что в свою очередь влечёт нарушение мыслительного процесса. Или, например, если программа вынуждает оператора проверять значения множества параметров, не имеющих прямого отношения к выполняемому процессу, то это тоже классифицируется, как избыточная информация. Вместо того, чтобы ответственно проверить необходимые значения, оператор тратит время на фильтрацию предоставленных ему параметров, что приводит к повышенной мыслительной нагрузке, и как следствие – ускоренному переутомлению.

Монотонность обуславливается недостаточным разнообразием в работе, или рабочем окружении; повторяемыми операциями или низкой сложностью задачи, особенно в течение длительного времени. Повышение разнообразия в работе, предоставление оператору работать в собственном темпе, разрешение делать перерывы в работе и настройка оптимальной освещённости (в идеале разрешить настройку пользователям), позволяет существенно снизить монотонность. Отсутствие раздражителей из-за повсеместной автоматизации процессов и снижение уровня вовлечённости операторов в производственный процесс приводят к монотонности и снижению внимательности. Удерживание внимания является ещё одной проблемой, поскольку снижение внимательности может привести к спаду производительности (например, несвоевременное реагирование на сигналы оповещений). Это может произойти всего за период от 10 до 20 минут. Желательно, чтобы система не требовала постоянного внимания оператора, а, по меньшей мере, предоставляла технические средства мониторинга и реагирования на важные события.

Также переутомление происходит, когда выполняемая работа требует рутинных, однообразных или похожих действий. Для предотвращения этого, функции ПО должны быть правильно распределены между человеком и машиной: машине полагаются простые и повторяющиеся задачи, а пользователю отводятся оформленные задачи, включающие в себя элементы планирования, выполнения и контроля.

Производственная безопасность в Евросоюзе регулируется двумя независимыми министерствами. Если речь идёт о товарах народного потребления, то применяются директивы, описанные в статьях 114/115. В соответствии с «Новым подходом» базовые требования к безопасности формируются в виде директив [22]. Технические спецификации соответствия продуктов основным требованиям выносятся в директивы и компонуются в виде упорядоченных стандартов ( [29] и [30] для производства). Сертификация по каким-либо стандартам добровольная и производитель в праве варьировать набор директив для соответствия тем или иным требованиям. Товары, выпущенные в соответствии стандартам, имеют преимущество в виде заведомо известного соответствия основным требованиям. С другой стороны, рабочие места не тестируются на соответствие каким-либо стандартам. Каждое государство в праве самостоятельно компоновать требования, основываясь на директивы минимального уровня безопасности из [31], не нарушая при этом главный нормативный документ. Национальная легализация нацелена на сотрудников и поэтому востребована в данной области.

Обе эти области имеют прямое отношение к проектированию и использованию ЧМИ. Во-первых, человеко-машинное взаимодействие подразумевает, интерфейс управления, который в свою очередь должен соответствовать [25] для того, чтобы операторы могли пользоваться им эффективно и безопасно, не подвергая угрозе своё здоровье и самочувствие. Стандарт [25] даёт наставления разработчику в вопросах анализа перечня и способов решения задач проектируемого интерфейса. Данная процедура способствует разработке и определению:

* перечня функций, выполняемых человеком в отношении машины;
* перечень сведений, сообщаемых машиной человеку;
* степень автоматизации управления;
* детали решения задач оператором.

Кроме того, сформированы 10 критериев хорошо спроектированных интерфейсов приводятся в качестве примеров для облегчения проектирования. Туда входят критерии, начиная с оценки уровня подготовки оператора и заканчивая оценкой коэффициента обратной связи для предотвращения избыточной многозадачности и переутомления. В документе в качестве образца [25] приводится процесс проектирования интерфейса сверлильной машины.

Дизайн интерфейса задач в свою очередь является каркасом, или внешней границей при проектировании интерфейса взаимодействия. ЧМИ чаще всего относят к интерфейсу взаимодействия, который должен проектироваться с учётом особенностей восприятия человеком информации. Иными словами, интерфейс, должен использовать все существующие органы чувств, способность оператора рассуждать и критически мыслить. Важно при этом соблюдать пределы человеческих возможностей. Если перед оператором стоит задача выключить прибор, то выключатель должен присутствовать в поле зрения, его размер и форма должны быть таковыми, чтобы данный элемент управления был воспринят на правильном уровне когнитивного взаимодействия. Внешний вид и форма элемента управления, в данном случае, будут иметь решающий фактор при принятии решения, а физические параметры отойдут на второй план. Сила, с которой должно быть произведено воздействие на элемент управления, а также отклик тоже играют немаловажную роль при человеко-машинном взаимодействии.

Для совершения операции человек сообщает команду машине посредством элементов управления, и анализирует результат своего воздействия, посредством наблюдения за дисплеями и различения звуковых сигналов, а также, наблюдая за работой самой машины.

### Внешний вид и расположение элементов управления

Европейский стандарт [30] содержит сведения о правильных видах актюаторов и элементов управления для различных задач. Производитель оборудования (или разработчик программы) при разработке должен следовать наставлениям из данного документа. Нельзя сказать, что рекомендации имеют жёсткий характер: документ не навязывает определённый элемент, а раскрывает критерии выбора более или менее подходящего варианта для отдельно взятого случая. Для того, чтобы элемент управления использовался эффективно, он должен отвечать критериям эргономики. Документ рассматривает следующие критерии.

Размеры элемента управление и расстояние до соседних элементов должно учитывать удобство захвата. В случае с применением курсоров типа «мышь», «кнюпель» или «трекбол» размеры будут одни, а применении сенсорного ввода, например, с помощью резистивного, ёмкостного или инфракрасного дисплея – другими.

Расположение элемента управления относительно оператора играет важную роль в вопросах эргономики. Дисплеи маленького разрешения, а также использование указывающих устройств с относительным позиционированием (такие как мышь, тачпад, трекбол) продиктовали в своё время такой способ размещения элементов управления на экране, когда их физические размеры крайне малы и расстояние между ними минимально. С повышением разрешающей способности дисплеев и появлением указывающих устройств абсолютного позиционирования (таких как сенсорный дисплей или стилус) позволили увеличить размеры элементов на экране, а также отступов между ними. С одной стороны, это было обусловлено сравнительно низкой точностью указывающих устройств (пользователю трудно указать пальцем в маленький предмет и при взаимодействии с одним элементом, должна быть уверена, что не среагируют окружающие его элементы). С другой стороны, при повышении разрешающей способности отображающего устройства, физический размер элементов уменьшается, ввиду увеличения плотности пикселей (обычно обозначается как PPI). С третьей стороны, возможность абсолютного позиционирования устранила необходимость в сокращении расстояния между элементами, поскольку пользователю не требуется преодолевать это расстояние с помощью указывающего устройства. Такие перемены приводят к серьёзным эргономическим проблемам при разработке универсальных интерфейсов – встают вопросы адаптивности. Компромисса бывает достичь очень трудно: пользователи с устройствами абсолютного позиционирования жалуются, что «не попадают» в элементы, а пользователи относительных жалуются на существенное увеличение лишних перемещений указывающего устройства.

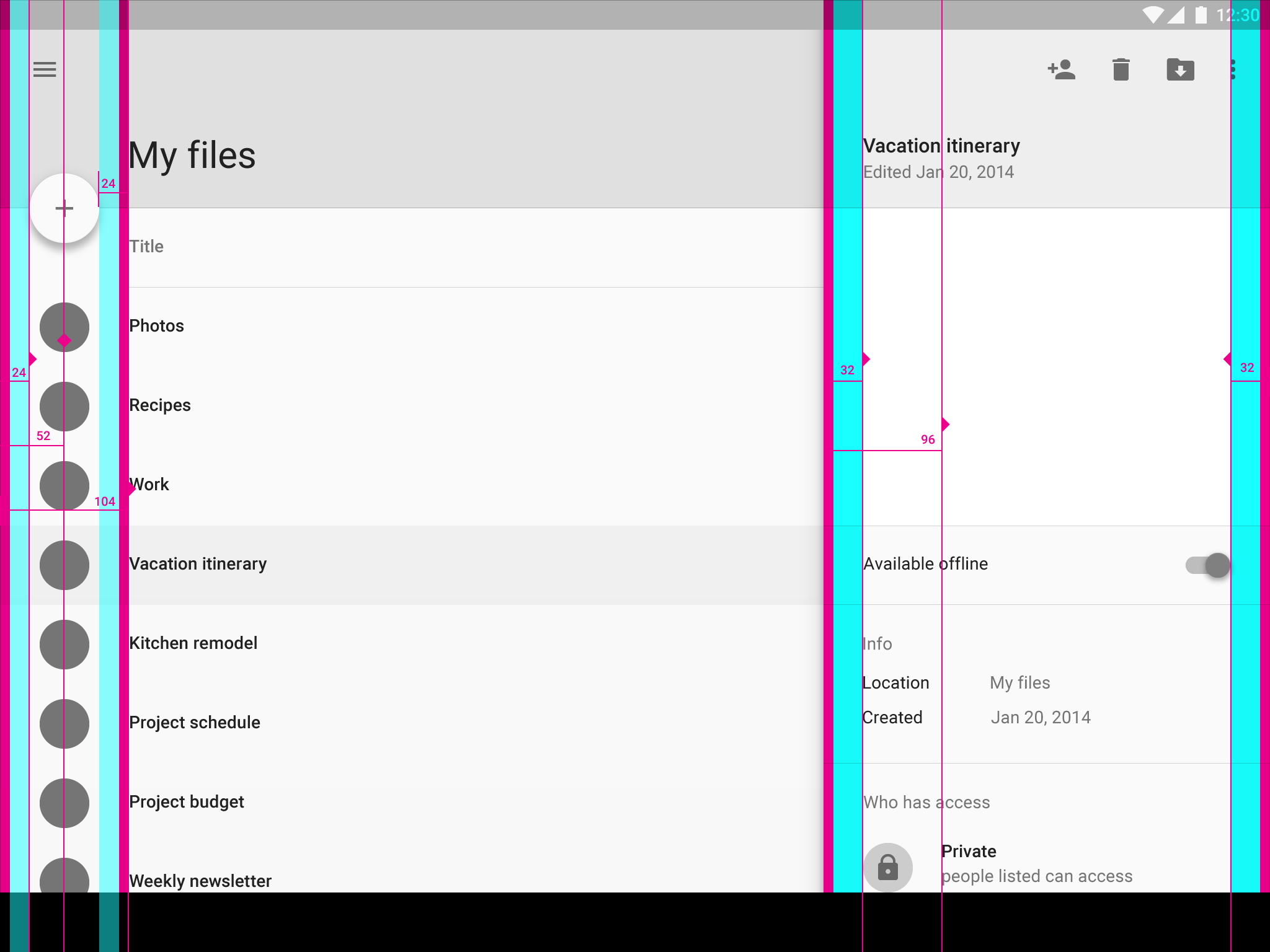


Рис.2.5. Графическое представление требований к сенсорному ЧМИ

Пользователь получает сведения о состоянии машины непосредственно с помощью органов чувств, либо косвенно – с помощью сигналов, шкал и дисплеев. Косвенная информация может быть передана визуально (например, с помощью сигнальных ламп, аналоговых и цифровых дисплеев), или звуковым способом (например, предупредительными сигналами, или снижением скорости работы машины). Также существуют способы тактильной передачи косвенных сигналов (например, с помощью вибромоторов).

Комбинируя все элементы системы как единое целое, проектирование эргономики подразумевает учёт cследующих характеристик человеческого тела [26]:

* дисплеи должны быть различимы оператору в естественном положении, без необходимости резких движений, в том числе поворотов головы и шеи;
* информация на дисплеях должна легко читаться, т.е. размер символов, контраст, разрешения и цвета должны способствовать восприятию;
* форма дисплея должна соответствовать характеру отображаемой информации и скорости её обновления;
* частота и громкость звуковых оповещений должны соответствовать важности событий, которыми они вызваны.

### Принципы описания «юзабилити»

Международный стандарт [13] описывает детали взаимодействия с пользователем посредством дисплея, что в западной литературе часто носит название «юзабилити». Далее представлен перечень актуальных принципов при проектировании юзабилити:

* соответствие решаемым задачам;
* информативность;
* управляемость;
* соответствие пользовательскому опыту;
* устойчивость к ошибкам;
* возможность персонализации;
* возможность изучения;
* способы представления информации.

Диалог соответствует решаемой задаче, если он эффективно взаимодействует с пользователем по пути решения поставленной задачи. Диалог должен ставить перед оператором только те вопросы, которые имеют непосредственное отношение к выполняемому действию. Одним из способов эффективного решения задачи пользователя – это формирование макросов или значений по умолчанию для повторяющихся случаев. Таким образом ликвидируется потребность в многократном вводе одних и тех же значений.

Диалог информативен, если каждый из вопросов моментально понятен пользователю из контекста на экране или разъяснение доступно в виде интерактивной справки». Диалог считается информативным, если, пользователю требуется минимальное знакомство с документацией. Перечень доступных действий должен быть понятен пользователю исходя из сведений, представленных на экране.

Диалог предоставляет управляемость, если пользователь в состоянии контролировать взаимодействие с системой в течение всей работы от начала и до достижения цели». Одним из главных примеров управляемости является наличие команд «отмена» или «назад», которые позволяют отменять действия или осуществлять возврат к предыдущей части диалога.

Диалог соответствует пользовательскому опыту, если он основывается на известных оператору сведениях о выполняемой задаче, общеобразовательных принципах, или общепринятых устоях. Примером данного принципа может быть интерфейс регулирования температуры. Если элемент управления расположен таким образом, что чем выше положение ползунка на шкале – тем вышеуказанная температура, то такой диалог соответствует общепринятым устоям. А если бы ползунок был перевёрнут, то это было бы нарушением.

Диалог устойчив к ошибкам, если, несмотря на очевидные ошибки во входных параметрах, желаемый результат всё равно может быть достигнут без необходимости или с минимальными правками. Ошибки должны быть представлены пользователю с целью их устранения». Советы по исправлениям, возможные альтернативы или даже автозамена способствуют удобству взаимодействия с ЧМИ. Кроме того, информативные сообщения об ошибках, с указанием того какая ошибка произошла и почему, лучше, чем просто сообщения о неудаче в ходе операции.

Диалог поддерживает персонализацию, если система допускает модификации под конкретные условия работы или навыки оператора». Можно утверждать, что персонализация имеет место, если пользователь может сменить язык, на котором выводится информация или подстроить интерфейс для противодействия недугам (например, увеличить шрифт при плохом зрении).

Диалог реализует возможность изучения, если этапы в нём сопровождаются обучением, завершив которое, оператор минимизирует время работы с ним. Диалог поддерживает обучение, если он содержит соответствующие обучающие стратегии, такие как сопровождение или примеры. В дополнение к этому, для предотвращения избыточной умственной нагрузки и, учитывая человеческие возможности по обработке информации, метод аналогии лучше, чем метод заучивания.

Принципы представления информации описаны в ISO 9241-12. Этот стандарт описывает принципы компоновки и размещения информации на экране, а также вопросы сочетания изображений и текста.

### Существующие способы оценки эргономики

Международный стандарт [32] даёт определение термину «юзабилити» как «степень эффективности и удовлетворённости определённой группы пользователей при использовании продукта для достижения цели в рамках заданного контекста. В другом документе говорится, что целью каждой программы является оценка её пользователями или экспертами на предмет соответствия заданному критерию, т.е. выявление степени эффективности и лёгкости освоения, а также удовлетворения человеческих потребностей.

К сожалению, привлечение достаточного числа пользователей для проверки всех особенностей взаимодействия с определённой системы, затратно как по времени, так и ресурсам. В статье описываются методики неформальной оценки эргономики ЧМИ с привлечением экспертов, как способ «сэкономить на пользователях». Автор считает, что оценка «юзабилити», это «набор экономичных способов найти проблемы эргономики».

Несмотря на то, что пользователи должны привлекаться на всех этапах разработки ПО, методики оценки особенно полезны на начальных этапах разработки. Наиболее популярные и часто применяемые методики – это эвристическое выполнение и когнитивный обход.

Эвристическое выполнение – это «метод обнаружения проблем «юзабилити» в ЧМИ, путём оценки интерфейса небольшой экспертной группой, обладающей опытом работы с системами подобного типа». Это малозатратный метод, поэтому он часто применяется в условиях ограниченного бюджета. Кроме того, привлечение экспертов даст более качественный анализ, чем пользователи без опыта.

Поскольку метод когнитивного обхода не требует законченного проекта и участия пользователей, он уместен на самых ранних этапах разработки. Обычно эксперты оценивают вероятность возникновения проблем с точки зрения пользователя. Во время выполнения, особое внимание уделяется «первому впечатлению» - первому использованию без предварительной подготовки. Для когнитивного обхода необходимы:

* описание пользователей и их уровня знания;
* описание задач, выполняемых системой;
* перечень правильных действий для выполнения задач.

Применение методов оценки недостаточно и не является заменой пользовательского тестирования. Наилучших результатов можно добиться, объединив оба метода, поскольку и оценка, и эмпирический подход позволяют обнаружить проблемы, скрытые при использовании этих методов по отдельности. Методика эмпирической оценки включает в себя протокол рассуждений «мысли вслух», интервью, опросы, а также лабораторные «юзабилити»-тесты.

Метод «мысли вслух» применяется особо часто во время «юзабилити»-тестов. Во время тестирования, участников просят высказывать свои мысли при выполнении задач. С помощью данного метода эксперты обнаруживают проблемы на практике и узнают общее мнение пользователей касательно дизайна программы.

### Технологии для людей с ограниченными возможностями

Многие человеческие свойства, особенно физиологические аспекты, меняются с возрастом. Сенсорная способность (зрение, слух, осязание) притупляется, снижается производительность сердечно-сосудистой системы. Эти недостатки, которые долгое время оставались эпицентром обсуждений демографического сдвига, противопоставляются другим характеристикам, которые, в свою очередь, улучшаются с возрастом. Такие характеристики напрямую связаны с жизненным опытом, такие как социальные навыки и навыки общения. Угасание некоторых характеристик можно замедлить тренировкой. Например, общеизвестно, что регулярная тренировка мышц позволяет поддерживать их производительность на высоком уровне, несмотря на старость. Иногда угасающая способность может компенсироваться опытом или приобретёнными навыками. В определённых случаях жизненный опыт может даже давать пожилым людям преимущество перед их молодыми коллегами.

Физические недостатки или снижение сенсорных способностей могут нивелироваться хорошо продуманным дизайном элементов ЧМИ. Высокий контраст и большие символы на экранах способствуют их распознаванию. Звуковые сигналы, характеризующие состояние машины, могут иметь настройку частоты и громкости, чтобы обеспечить одинаковую различимость, как молодыми, так и пожилыми людьми. Хорошо спроектированный ЧМИ, благодаря своим эргономическим изысканиям, позволяет людям всех возрастных категорий работать одинаково эффективно, без страданий или жалоб на здоровье.

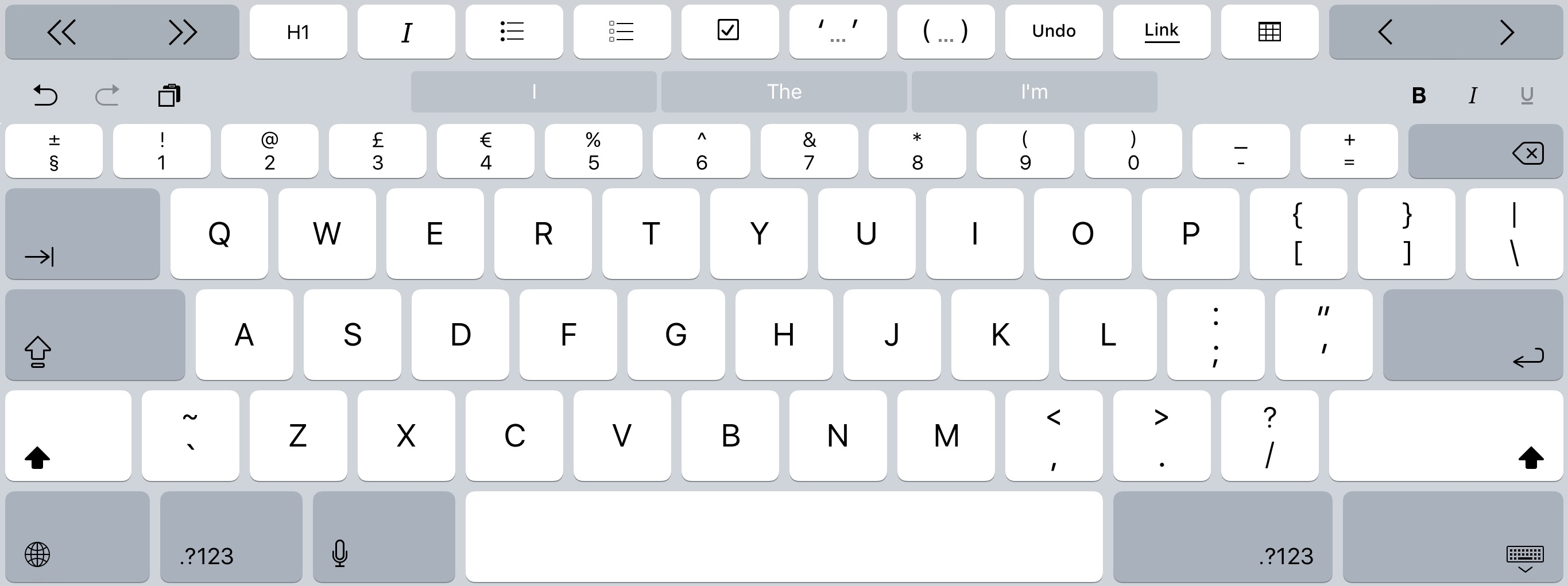


Рис.2.6. Вспомогательные технологии для сенсорных ЧМИ

Более серьёзные физические недостатки могут привести к неэффективному труду или к увечью. Принципы проектирования изделий, с учётом людей с ограниченными возможностями, относятся к эргономике доступности. В данном контексте, учитывая технику безопасности при производстве, «доступность» не означает, что кто угодно, без всяких исключений, должен иметь возможность использовать оборудование без применения дополнительных мер по настройки и модификации органов управления. Козловой кран не допускает управление людьми, не прошедшими предварительного обучения. Это противоречило бы самой сути безопасности. Не следует всегда рассматривать обывателя как основного пользователя системы.

## Математическая модель численной оценки эргономических качеств на основании представленных критериев

Алгоритм получает сведения о данных, предоставляемых пользователем программе и компонует панель оператора из стандартных элементов управления (текстовое поле, ниспадающее меню и т.д.). Оценка качества полученного интерфейса производится путём минимизации функции штрафа. Штраф оценивается как функция накопления ошибок компоновки, приводящих к когнитивному диссонансу у пользователя. Для формирования понятия ошибок компоновки использовались метрики GOMS, а именно:

1. Ввод текста или нажатие клавиш
2. Наведение курсора на объект
3. Перемещение руки на клавиатуру
4. Проведение отрезка на сетке

В общем виде эта метрика имеет вид:

, где

- ввода текста или нажатие определённых клавиш;

- наведение курсора на объект;

- перемещение руки на клавиатуру;

- проведение отрезка на сетке.

Элемент управления представлен множеством

, где

– x-координата расположения на экране;

– y-координата расположения на экране;

– занимаемая экранная площадь.

Тип элемента управления представлен функцией

, где

для ввода с клавиатуры значение 0;

для остальных случаев значение 1.

Дополнительно учитывалось понятие «Линии восприятия». Человек воспринимает естественный порядок элементов сверху-вниз и слева-направо. Нарушение этого порядка приводит к тому, что человек начинает визуально перебрать все элементы на панели в поисках следующего. Формализовано это выглядит так:

, где

при значение 0;

для остальных случаев значение 1.

Кроме всего вышеперечисленного, требуется, чтобы элементы интерфейса наиболее эффективно занимали предоставляемую плоскость экрана. Выражается это следующим образом:

, где

- элемент управления ;

- доступная площадь экрана.

Подытоживая вышесказанное, получается формула штрафа:

, где

– количество элементов управления;

– расстояние между элементами;

- площадь элемента

Алгоритм минимизирует функцию штрафа путём перестановки элементов управления и определения их инвариантов. Одно и то же значение может задаваться различными элементами, например, ползунком и текстовым полем. Если это значение располагается между текстовыми элементами, то разумно выбрать для него текстовый элемент управления, чтобы пользователь мог переключаться между ними с помощью одной клавиши Tab, не прибегая к использованию мыши (что в свою очередь добавит к штрафу коэффициенты , и ). Наиболее близкой к данной задаче является классическая задача динамического программирования «Задача о рюкзаке». В общем виде задачу можно сформулировать так:

«Из заданного множества предметов со свойствами «стоимость» и «вес» требуется отобрать подмножество с максимальной полной стоимостью, соблюдая при этом ограничение на суммарный вес».

Аналогия просматривается следующим образом. В качестве суммарного веса выступает обратная функция штрафа. Задача о рюкзаке максимизирует вес, задача о компоновке минимизирует штраф.

Задача о рюкзаке выбирает предметы, а задача о компоновке выбирает инварианты элементов управления. Задача о ранце относится к классу NP-полных, что означает, что для неё не существует полиномиального алгоритма решающего её за конечное время. Традиционно для решения этой задачи используют либо метод полного перебора, либо методами динамического программирования.

Однако для своей задачи я предлагаю использовать генетический алгоритм. Структура хромосомы имеет вид:

, где

– координаты элемента в сетке

– тип элемента

– общее число всех элементов

1. Оценка сложности

Оценим сложность решения задачи о рюкзаке методами динамического программирования.

Пусть вес каждого предмета является целым неотрицательным числом. Тогда для решения задачи необходимо вычислить оптимальные решения для всех , где — заданная грузоподъемность. Определим как максимальную ценность предметов, помещаемых в рюкзак грузоподъёмностью . Составим рекуррентные формулы для :

Где – ценность и вес -го предмета соответственно, а максимум из пустого множества следует считать равным нулю. Для решения данного функционального уравнения достаточно вычислить все значения , начиная с и до .

Так как на каждом шаге необходимо найти максимум из предметов, алгоритм имеет вычислительную сложность . Поскольку может экспоненциально зависеть от размера входных данных, алгоритм является псевдополиномиальным. Поэтому эффективность данного алгоритма определяется значением . Исходя из вида формулы , сложность W оценивается как . В итоге имеем

Оценим сложность решения задачи о компоновке генетическим алгоритмом.

На каждом этапе эволюции должно быть вычислено значение фитнес-функции каждой особи в популяции, после чего особи должны быть упорядочены по значениям их приспособленности. В данном случае фитнес-функцией является обратная функция штрафа, что совпадает с функцией веса одного предмета в задаче о рюкзаке. Исходя из вида формулы , сложность равна . Сложность сортировки элементов популяции оценивается как . Принимая во внимание тот факт, что в процессе эволюции должно быть найдено несколько поколений, получим оценку вычислительной сложности , что меньше, чем в методе динамического программирования.

Выводы по главе II

1. Проанализированы актуальные эргономические требования:
   * ограничения мыслительной деятельности;
   * модель памяти человека;
   * внешний вид и расположение элементов управления;
   * принципы описания юзабилити;
   * существующие способы оценки эргономики;
   * технологии для людей с ограниченными возможностями.
2. Разработана математическая модель численной оценки эргономических качеств на основании рассмотренных критериев и произведена оценка эффективности предполагаемого алгоритма оптимизации.

# Разработка модифицированной методики проектирования ЧМИ

## Разработка UML-диаграмм для определения набора команд, доступных оператору

Первым шагом модифицированной методики является составление UML диаграммы вариантов использования. Варианты использования составляют важную часть UML документации. Они позволяют дать целостное представление о функциях, выполняемых системой, тем самым формируя список требований. Также их можно использовать для обсуждения вопросов при проектировании между разработчиками, архитекторами, заказчиками и руководителями. Обычно, на базе вариантов использования системы впоследствии строятся тесты, которые служат критерием верификации ПО на предмет соответствия техническому заданию. Варианты использования легки для понимания, в том числе людьми далёкими от информационных технологий. Варианты использования позволяют дать целостное видение системы, при этом, не вдаваясь в подробности реализации. С другой стороны, ошибки, допущенные при подготовке вариантов, могут иметь далеко идущие последствия. Они могут обнаружиться не только на этапе проектирования, но и на этапе программной реализации. Тем самым, возможность формального анализа диаграмм, может быть эффективным средством против этого. Кроме того, выводы, полученные в ходе формального исследования, позволяют проводить верификацию выбранных сущностей в модели системы. Первичный продукт анализа требований в формате вариантов использования может быть представлен в двух формах:

* в виде графических диаграмм, перечисляющих непосредственно варианты использования и актёров (обычно это пользователи и внешние системы), а также отношения между ними;
* в виде набора сценариев, описывающих потоки данных и событий, с точки зрения каждого из пользователей.

Несмотря на то, что диаграммы должны соответствовать строго формализованному синтаксису, существуют определённые техники, позволяющие дополнить описание поведения системы. Они включают неформальный естественный язык, ограниченный естественный язык, диаграммы действий или схемы последовательности сообщений. Варианты использования также применяются для формальной верификации требований на начальной фазе проектирования системы, что позволяет снизить расходы на разработку последующих фаз на протяжении всей разработки. Существуют два основных метода, которые применяются для формальной верификации вариантов использования. Первый из них носит название «исследование состояний», т.е. формируется как перечень достижимых состояний системы, которые впоследствии анализируются. Такой метод представляет собой подход к проверке модели. Второй метод относится к символьным выражениям, с применением математической логики и языка высказываний, что позволяет проводить семантический анализ свойств системы. Использование диаграмм вариантов использования в контексте формальной верификации и анализа уже активно используется. В работе [33], рассматривается подход, в котором особый упор делается на взаимоотношения между актёрами и вариантами использования, через т.н. «контракты». В работе [34] рассматривается пример валидации полученных диаграмм. Дуализм представления, а также неформальное описание сценариев порождает определённые трудности при оценке поведения всей системы в целом и валидации целостности между различными диаграммами и сценариями. Существуют исследования по переводу сценариев в формализованный вид. Работа [35] описывает примеры перевода вариантов использования в конечные автоматы. Стоит учитывать, что в методике отсутствует симметричный подход, который отслеживал бы связи между вариантами и обновлял бы отношения в моделях.

Суть модификации, описываемой в данной работе, заключается в использовании графического подхода к составлению диаграммы вариантов использования программы на начальном этапе проектирования. Каждый вариант использования потенциально требует графического представления для взаимодействия с пользователем. На этом этапе, анализируя всевозможные варианты можно оценить необходимость в разработке различных интерфейсов для отдельных случаев. Зачастую, среди всех представленных вариантов можно выделить схожие, а именно те, которые находятся в отношении «расширяет». Такие варианты могут быть представлены одним интерфейсом с дополнительными опциями. Например, вариант «фасовка сыпучего вещества» может быть объединён с вариантом «фасовка жидкости». В итоге они будут выполняться через один и тот же ЧМИ, но с дополнительными аргументами в случае с жидкостью в силу технических ограничений, таких как минимально допустимый объём и максимально допустимая скорость фасовки.

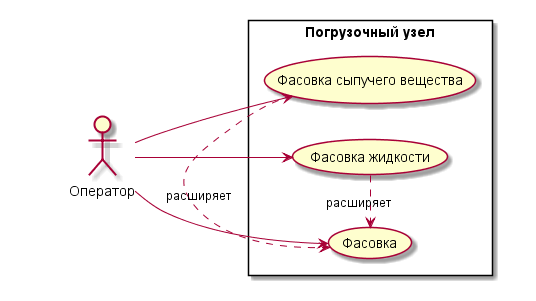


Рис.3.1. UML-диаграмма фасовочного узла

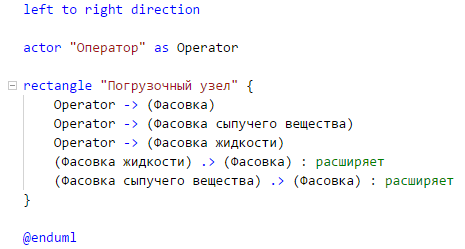


Рис.3.2. Формализованное представление вариантов действия

## Разработка UML-диаграмм последовательности действий для определения порядка следования элементов управления в макете

Вторым этапом предлагаемой методики является разработка последовательностей действий, приводящих к желаемому результату. Для этого применяется ещё один вид UML диаграмм – диаграмма последовательности. Такие диаграммы используются главным образом для того, чтобы показывать порядок взаимодействий между объектами. Точно также, как и в случае с диаграммой классов, считается, что данный вид документации предназначен исключительно для описания программного кода. Однако всё больше и больше других отраслей находят эти диаграммами полезными для описания модели взаимодействия бизнес-объектов. Помимо описания уже существующих решений, такие диаграммы активно применяются при планировании, для обсуждения разрабатываемой модели взаимодействия. Благодаря этой форме представления информации, специалисты могут ещё на этапе планирования создать формализованное описание процессов между объектами. Зачастую временная диаграмма позволяет более подробно показать взаимодействие, изначально представленное на схеме вариантов использования.

Основным применением диаграмм последовательности является вывод изначальных требований, обычно представленных в виде вариантов использования, на новый уровень понимания. Обычно одна диаграмма вариантов использования порождает несколько диаграмм последовательности. Помимо описания разрабатываемых систем, такие схемы уместны при документировании уже существующих систем. Данный вид документации бывает крайне полезен при передаче системы другому человеку или организации.

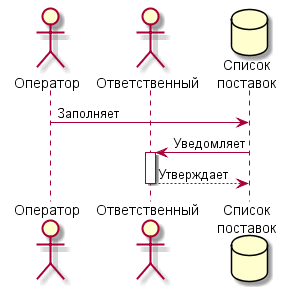


Рис.3.3. Диаграмма последовательности с асинхронной операцией

Главная цель диаграммы последовательности – дать определение последовательности действий, приводящих к желаемому результату. Акцент делается не столько на сообщениях между объектами, сколько на их очерёдности; тем не менее сами сообщения всё равно указываются, хотя и в сокращённом виде. Диаграмма рассматривается по вертикали и горизонтали. По вертикали ведётся отсчёт времени и наличие сообщений от конкретного объекта. По горизонтали слева-направо указывается получатель сообщения.

При создании диаграммы последовательности сверху размещаются объекты, жизненный цикл которых будет описываться ниже. Линии жизни описывают либо отдельные роли, либо экземпляры классов, участвующих в информационном обмене. Линии жизни отображаются как прямоугольники с отходящей от их нижней границы пунктирной линией. Название линии указывается внутри прямоугольника.

Стандарт UML предписывает следующую нотацию линий жизни: «Название экземпляра : Название класса». Допускается объявление безымянных линий. В таком случае остаётся символ двоеточия и название класса.

Первое сообщение на диаграмме всегда указывается сверху и обычно располагается с левой стороны диаграммы для удобства чтения. Последующие сообщения идут ниже. Передача сообщения указывается как сплошная линия со стрелкой от передающего объекта к получающему. Название сообщения или метода указывается над стрелкой. Сообщение, направленное к объекту, обозначает, что получатель реализует указанную операцию или метод. Ответные сообщения являются опциональными по спецификации UML, однако в данной методике они обязательны. Это вызвано тем, что по критериям эргономики, пользовательский интерфейс должен реагировать на воздействия пользователя, т.е. должна быть обеспечена обратная связь. Использование ответного сообщения позволяет на диаграмме указать характер и направление обратной связи.

При разработке ЧМИ иногда возникает необходимость указать связь объекта с самим собой. Это необходимо в тех случаях, если использование отдельного элемента управления вызывает подпрограмму, которая, в свою очередь, может быть вызвана другим элементом управления. Стандарт UML допускает такой вид взаимодействия.

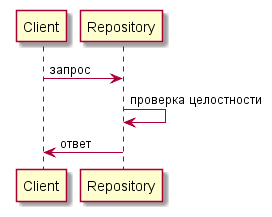


Рис.3.4. Замыкание объекта

При разработке многопоточных программ, возникает необходимость в отображении на схеме асинхронных операций. Например, в ситуации, когда оператор запускает долгую задачу, система должна отобразить смену состояния и продолжить актуализацию показаний на дисплее. В диаграмме последовательности это будет выглядеть как асинхронная операция. Отличие данного вида сообщения составляет видоизменённый наконечник стрелки.

Графический пользовательский интерфейс позволяет пользователю нелинейно взаимодействовать с ПО. Это означает, что взаимодействие невозможно описать в виде непрерывной диаграммы. Для решения данной проблемы, используется возможность UML применять опциональные блоки. С помощью опциональных блоков можно описать сложный интерфейс, допускающий различные пути при работе с программой.

## Разработка алгоритма выбора навигационной парадигмы

Третьим этапом проектирования ЧМИ по данной методике идёт определение навигационной парадигмы. Если проектируемое устройство обладает обширным функционалом, или количество элементов управления настолько велико, что их имеет смысл визуально разделить, то интерфейс превращается в коллекцию страниц. Под навигацией, в таком случае, подразумевается процесс перехода между страницами, либо между разделами одной страницы. Навигация является ключевым фактором «юзабилити», поскольку именно это свойство определяет то, как оператор обнаруживает тот, или иной функционал. Это очень важный элемент проектирования и бывает довольно сложно разработать его с первого подхода. Сложность состоит в том, что на сегодняшний день существует обилие навигационных парадигм, позволяющих предоставить доступ к одним и тем же функциям, но разными путями. Можно заставить пользователя пролистывать перечень страниц в поисках нужной, а можно сформировать меню, которое обеспечит возможность прямого доступа к интересующей странице. Также можно расположить все элементы на одной странице, но предоставить алгоритм фильтрации, с помощью которого можно скрыть лишние.

Поскольку не существует единой универсальной парадигмы, в данной методике предусмотрен алгоритм выбора одной из наиболее подходящей в конкретном контексте.

Существуют три базовых принципа правильной навигации:

* целостность: соответствие ожиданию пользователя;
* простота: отсутствие лишних элементов управления;
* ясность: обеспечение прозрачных путей и вариантов.

Навигация должна соответствовать ожиданиям пользователей. Использование типовых элементов управления, с которыми пользователь уже работал в других программах, а также следование стандартным соглашениям о применении значков, расположений и оформления, позволяет разработать предсказуемую и интуитивно-понятную навигацию для пользователя.

Сокращение количества элементов навигации снижает мыслительную нагрузку на пользователя. Обеспечение быстрого перехода к важным для оператора разделам и скрытие менее важных пунктов повышает скорость взаимодействия человека с машиной. Существует два основных вида навигации: плоская и иерархическая.

В плоской модели все страницы сосуществуют рядом друг с другом. Пользователь переключается между ними в любом порядке. Соответствие перечисленным ниже критериев говорит о пригодности данной парадигмы в конкретном ЧМИ:

* допускается просмотр страниц в любом порядке;
* страницы имеют явные отличия в макете и не имеют зависимостей между собой;
* в группе менее семи страниц (когда страниц больше, то нарушается одно из базовых правил эргономики «три плюс-минус два» и пользователю становится затруднительно определить расположение других страниц относительно своего текущего положения).

Рис.3.5. Плоская навигация

Другая рассматриваемая навигационная парадигма в данной методике – иерархическая структура. В такой модели страницы образуют древовидную структуру. Каждая дочерняя страница имеет как минимум одного родителя, но каждый родитель может иметь более одного потомка. Чтобы попасть на дочернюю страницу, пользователь в начале должен посетить родительскую.

Иерархические структуры полезны в тех случаях, когда функционал ПО обилен и занимает большое число страниц. Недостатком данной парадигмы выступает излишняя сложность – чем глубже структура, тем больше лишних взаимодействий совершает пользователь перед тем, как попасть на целевую станицу. Критерии применения иерархической структуры следующие:

* перемещение по страницам допускается только по заданной схеме;
* между страницами существует логическая связь;
* в группе более семи страниц.

Рис.3.6. Иерархическая навигационная структура

Данная методика подразумевает, что одна программа может содержать в себе несколько парадигм сразу. После анализа вариантов использования и построения диаграмм последовательности, формируется перечень функциональных страниц. В этот момент и происходит определение навигационной парадигмы. В потребительских программах обычно корневые страницы образуют плоскую навигацию, а дочерние – иерархическую. В промышленном ПО ситуация обстоит наоборот. Это вызвано тем, что при использовании промышленного оборудования сначала проводятся этапы инициализации, которые задают дальнейший режим работы прибора. После того, как режим работы определён, начинается долгий цикл однообразного выполнения. В этом режиме и применяется плоская навигация.

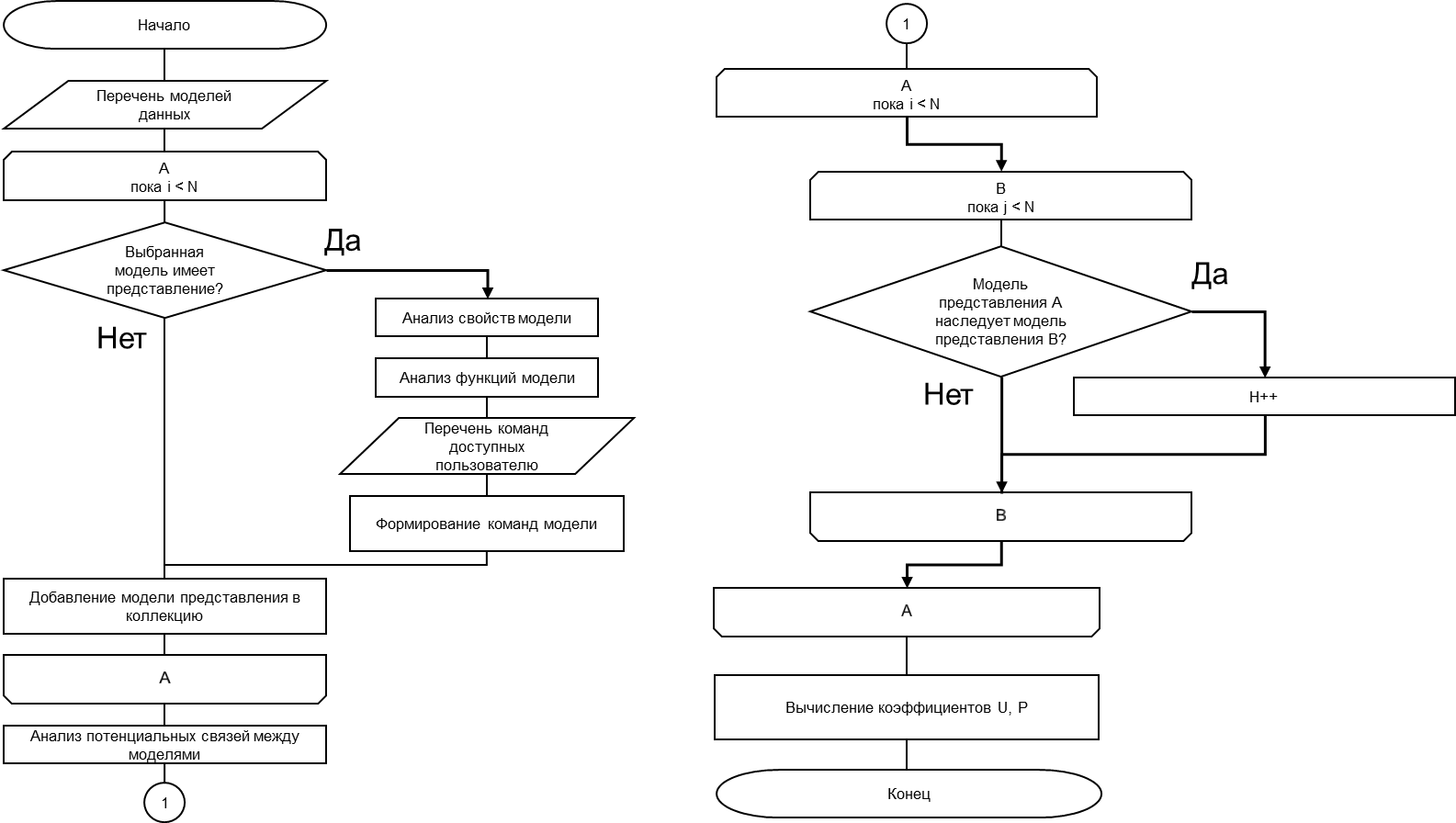


Рис.3.7. Модифицированный алгоритм реализации навигационной парадигмы

## Разработка алгоритма выбора элементов управления

После определения навигационной парадигмы и перечня команд, доступных оператору, необходимо произвести выбор элементов управления, олицетворяющие те самые команды.

Элементы управления – это графические объекты на экране компьютера, позволяющие оператору взаимодействовать с ПО [36]. Также их иногда называют «виджетами». Элементами управления являются:

* Кнопки
* Флажки
* Ссылки
* Панели вкладок
* Диаграммы
* Окна

Приведённый выше список далеко не полный: отдельные графические среды содержат разнообразные наборы элементов управления. Некоторые из них позволяют разработчику самостоятельно создать элемент управления, основываясь на графических примитивах и событиях. Использование готовой библиотеки элементов имеет очевидные преимущества. Готовая библиотека обеспечивает согласованный внешний вид элементов управления. Кроме того, графические библиотеки могут содержать в себе вспомогательные компоненты для людей с ограниченными возможностями, такие как виртуальная клавиатура, экранный диктор, цветовые схемы с высокой контрастностью и т.д. Немаловажным параметром выступает производительность библиотеки: иногда привлекательный внешний вид негативно сказывается на времени отклика пользовательского взаимодействия. Дополнительным фактором, влияющим на выбор библиотеки, является инструментарий разработки, поставляемый вместе с ней. В то время как одни наборы предполагают ручное кодирование макета, другие оснащены визуальным редактором, который позволяет в режиме реального времени позиционировать элементы на экране.

Элементы управления содержат атрибуты, методы и значения, управляющие их поведением и внешним видом. Разработчик может разместить элемент управления «флажок» в изначально «взведённом» положении. Этого может требовать конкретный вариант использования, когда пользователю нужно показать, что данный параметр рекомендуется к применению. Для этого разработчик изменяет значение атрибута. Стоит учесть, что это не лишает пользователя выбора: он по-прежнему может снять отметку, если посчитает, что в его ситуации данный параметр не нужен.

В рамках данной работы, элементы управления рассматриваются в общем плане, т.е. без привязки к какой-либо конкретной библиотеке. Элементы управления подразделяются по следующим критериям:

* направление потока информации (ввод/вывод);
* тип вводимых данных (текст/число/специализированное значение)
* диапазон значений (ограниченный/неограниченный);
* ожидаемая длина вводимых/отображаемых сведений;

Первым шагом выбора элемента управления определяется направление информации. Если пользователь вводит данные в программу, то это элемент ввода, если наблюдает за сведениями – то это элемент вывода. С программной точки зрения это поле типа «String». Данный алгоритм определяет направление информации по наличию публичного метода чтения (Getter) и устанавливающего метода (Setter): при наличии публичного метода чтения поле считается доступным для вывода. При наличии публичного устанавливающего метода – для ввода. Довольно часто бывает, что программа демонстрирует пользователю рекомендуемое значение параметра. Тогда элемент управления получается комбинированным. В таком случае он сочетает в себе как характеристики элемента ввода, так и вывода.

Простейшим элементом ввода в таком случае является «Текстовое поле». Текстовое поле представляет собой графический примитив в форме прямоугольника с расположенным в нём курсором. Допускается ввод любых символов с клавиатуры. Опционально текстовое поле может быть дооснащено валидацией вводимых параметров, кнопкой очистки, а также водным знаком, поясняющим назначение поля.



Рис.3.8. Текстовое поле

Простейшим элементов вывода является «Надпись». Надпись может отображать как статическое, так и динамическое содержимое программы с помощью разнообразных шрифтов, стилей и эффектов. Обычно ограничение на доступные варианты отображения накладывается либо используемой библиотекой элементов, либо графической подсистемой компьютера.

Вторым шагом при выборе категории виджета следует определение типа вводимых данных. Текстовое поле позволяет вводить как строки, так и числа; однако для ввода чисел предусмотрены специальные элементы ввода. В программах числа, задаваемые пользователем, чаще всего подразделяются на целые (Integer) и дробные (Float, Double, Decimal). Для целых чисел применяется элемент «Двунаправленный счётчик». Особая проблема возникает при вводе чисел с десятичным разделителем: в разных региональных настройках этот символ отличается. Например, в стандартной русской локализации «ru-ru» это запятая, в латинской «en-us» — это точка. Стандартная IBM-совместимая клавиатура содержит блок цифр «Numpad» с разделителем, соответствующим текущей выбранной раскладке клавиатуры. Однако при вводе чисел в раскладке, не соответствующей текущим языковым параметрам, происходят ошибки: либо десятичный разделитель не печатается вообще, либо система сообщает об ошибке. Таким образом, при вводе десятичных дробей необходимо использовать специализированный элемент управления, который автоматически подставляет корректный символ-разделитель независимо от выбранной раскладки.

Третьим шагом выбора элемента управления следует определение диапазона значений. Данная классификация в большей мере относится к элементам ввода. Если пользователь вправе вводить произвольные значения в программу, то текстовое поле идеально подходит для данного сценария. Однако если ввод ограничен фиксированным набором значений, то в данном случае следует использовать элемент управления «Выпадающий список», «Флажок» или «Радиокнопка». Выпадающий список получает перечень доступных вариантов значения поля и демонстрирует их пользователю в виде текстового поля, из которого по щелчку раскрывается список доступных элементов. Другим представлением выбора является радиокнопка. Это группа элементов управления, которая отображает все доступные варианты одновременно. При этом напротив одного из них пользователь вправе установить отметку, делающую выбранный пункт активным. Частным случаем выбора является «флажок». Он позволяет сделать выбор одного из двух доступных вариантов. Доступные варианты значения на программном уровне определяются как поля типа Перечисление (Enumeration) или сочетание полей одного типа массив – значение. В случае с «флажком» это поле типа Логическое (Boolean).

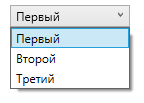


Рис.3.9. Выпадающий список



Рис.3.10. Флажок

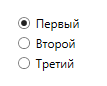


Рис.3.11. Радиокнопка

При вводе дробных чисел из диапазона применяется элемент управления «Ползунок». Данный виджет представляется в виде прямоугольника, перемещаемого графическим манипулятором вдоль линии, олицетворяющей диапазон значений. Ползунок может располагаться как горизонтально, так и вертикально, в зависимости от контекста задачи.



Рис.3.12. Ползунок

Наконец, финальный этап выбора элемента управления основывается на ожидаемой длине вводимых сведений. Если предполагается ввод текста более одной строки, то виджет «Текстовое поле» в таком случае неуместен. Необходимо использовать «Текстовое поле».

## Разработка генетического алгоритма компоновки операторской панели с целью минимизации функции штрафа.

Формирование макета операторской модели представляет собой заключительный этап модифицированной методики проектирования ЧМИ. Суть данного этапа заключается в оптимизации эргономических качеств страницы панели путём минимизации функции штрафа.

Генетическим алгоритмом называется метод решения как ограниченных, так и неограниченных задач оптимизации, основанный на естественном отборе – основополагающем процессе биологической эволюции. Генетический алгоритм многократно модифицирует популяцию решений. На каждом шаге генетический алгоритм выполняет отбор индивидов из текущей популяции на роль родителей и использует их для формирования следующего поколения. Через успешные поколения, популяция «развивается» в сторону оптимального решения. Генетические алгоритмы хорошо подходят для решения многих задач оптимизации, в частности тех, которые плохо вычисляются стандартными алгоритмами, например, недифференциируемые, стохастические, либо с высокой нелинейностью [37]. Ещё генетические алгоритмы могут применяться при решении задач на смешанное целочисленное программирование, когда значение компонент ограничено диапазоном целых чисел.

На каждом шаге генетические алгоритмы руководствуются тремя основными правилами при создании новой популяции из текущей:

* правила отбора (селекции) – описывают законы отбора индивидов, впоследствии называемых родителями, участвующих в формировании следующего поколения;
* правила рекомбинации (кроссинговера) – описывают закон перестановки свойств родительских индивидов при формировании потомка;
* правила мутации – описывают закон случайных изменений, в свойства родителя.

Генетические алгоритмы имеют характерные отличия от традиционных алгоритмов оптимизации, представленные в **Ошибка! Источник ссылки не найден.**.

Таблица 3.1. Характерные отличия генетических алгоритмов

|  |  |
| --- | --- |
| *Классический алгоритм* | *Генетический алгоритм* |
| Формирует точку на каждой итерации. Последовательность точек является аппроксимацией оптимального решения. | Формирует последовательность точек на каждой итерации. Лучшая точка в популяции является аппроксимацией оптимального решения. |
| Следующая точка последовательности определяется детерминированным вычислением. | Следующая популяция вычисляется алгоритмом с применением алгоритма генерации случайных чисел. |

Под фитнес-функцией в генетических алгоритмах подразумевается оптимизируемая функция. В традиционных алгоритмах оптимизации данная функция называется целевой. В процессе работы алгоритма происходит минимизация фитнес-функции.

Индивидом называется любая точка, в которой можно вычислить значение фитнесс-функции. Значение фитнесс-функции в конкретной точке называют значением приспособленности. В частности, для фитнесс-функции вида

Вектор , длина которого является число переменных в задаче, называется индивидом. Приспособленность индивида вычисляется как . Индивиды иногда называют геномом, а элементы вектора индивида генами.

Популяцией называется массив индивидов. Например, если размер популяции 100, а число переменных в фитнес-функции составляет 3, то популяция представляет собой матрицу 100х3. Один и тот же индивид может быть представлен в матрице более одного раза.

На каждой итерации генетический алгоритм выполняет серию вычислений нал текущей популяцией для формирования новой. Каждая последующая популяция носит название поколения.

Разнообразие вычисляется как среднее расстояние между индивидами в популяции. Считается, что у популяции высокое разнообразие, если среднее расстояние между индивидами большое; иначе – низкое. Разнообразие играет важную роль в генетических алгоритмах, поскольку оно позволяет расширить пространственный диапазон поиска.

Приспособленность индивида вычисляется как значение функции приспособленности для него. Поскольку в данной работе задачей стояло найти минимум функции, то наилучшем значением приспособленности считается наименьшее значение среди всех особей популяции.

Для создания следующего поколения генетический алгоритм осуществляет отбор определённых индивидов в текущей популяции, называемых родителями и использует их для создания индивидов следующего поколения, называемых потомством. Обычно алгоритм выбирает на роль родителей те особи, что имеют наилучшую приспособленность.

Ниже представлены основные этапы работы алгоритма.

1. Работа алгоритма начинается с формирования начальной популяции случайным образом.
2. Затем алгоритм создаёт последовательность новых популяций. Для создания каждой последующей выполняются следующие шаги.
   1. Вычисляется значение фитнес функции для каждого индивида. Эти значения называются грубым значением функции приспособленности.
   2. Вычисленные значения нормируются для приведения в более пригодный диапазон. Нормированные значения носят название ожиданий.
   3. Производится отбор особей, называемых родителями, на основе ожиданий.
   4. Несколько индивидов в текущей популяции с наименьшим значением функции приспособленности называются элитой. Элита переносится в следующее поколение.
   5. Осуществляется формирование следующего поколения. Потомки формируются либо путём мутаций – случайных изменений в родителе, либо путём кроссинговера – рекомбинации свойств пары родителей.
   6. Текущая популяция заменяется на следующее поколение.
3. Алгоритм останавливается по достижении заданных критериев остановки.

Алгоритм начинает работу с создания случайной начальной популяции. В данной методике сначала формируются 20 особей. Если предварительно известно приблизительное расположение минимума функции, то начальное поколение должно формироваться рядом с ним. В данном случае, очевидно, что упорядоченное размещение элементов управления в соответствии с логическим ходом их заполнения способствует снижению функции штрафа. Поэтому начальное поколение представляет собой столбец из элементов, упорядоченных в порядке взаимодействия. Кроме того, в некоторых системах (в частности, на мобильных устройствах) элементы располагаются в один столбец, поэтому данная расстановка имеет довольно хорошую приспособленность.

На каждой последующей итерации, генетический алгоритм использует текущую популяцию, чтобы создать потомство, из которого будет сформировано следующее поколение. Алгоритм выбирает группу индивидов в текущем поколении, называемом родителями, которые предоставляют свои гены – элементы их векторов потомкам. Алгоритм обычно выбирает в качестве родителей те индивиды, значение функции приспособленности которых минимально. На данном этапе формируется три типа потомков для следующего поколения [38]:

* элитные – это индивиды, приспособленность которых лучшая в своём поколении. Эти особи автоматически переходят в следующее поколение;
* комбинаторные – это потомки, образованные из генов пары родителей;
* мутационные – это потомки, образованные в процессе мутаций единственного родителя.

Алгоритм формирует комбинаторных потомков путём комбинирования пар родителей текущего поколения. В каждой координате вектора-потомка функция кроссинговера случайным образом определяет вхождение, т.е. ген одной и той же координаты у одного из родителей и присваивает его потомку. Для задач с линейным ограничением, потомок представляет собой взвешенное среднее родителей. Существуют следующие операторы выбора родителей:

* панмиксия;
* инбридинг;
* аутбридинг.

Панмиксией в генетике называют процесс произвольного скрещивания. В панмиксической популяции все индивиды являются потенциальными партнёрами [39]. Это означает, что не существует каких-либо ограничений: ни географических, ни поведенческих, что делает возможными любые рекомбинации. В генетике произвольное скрещивание подразумевает отсутствие физических, генетических и социальных преград. Другими словами, на процесс слияния не влияют факторы внешней среды, что делает вероятность участия каждого индивида в процессе скрещивания равновероятным. Произвольное скрещивание имеет место в условиях недостаточного естественного отбора. В природе приспособленность организмов производит селекцию до скрещивания.

С точки зрения живой природы панмиксия означает возможность скрещивания особей популяции без ограничений. Такое происходит, когда особи способны свободно перемещаться в пределах своей среды обитания и совокупляться с другими особями популяции. Чтобы подчеркнуть важность данного способа, рассмотрим различные конечные популяции одного и того же вида, изолированных определённой естественной преградой. В ходе прогресса, естественный отбор и дрейф генов образуют в популяциях генетическое разнообразие, что сделает каждую из них по-своему уникальной. Однако, если разделяющий фактор исчезнет до того, как это произойдёт (например, в лесу проложат дорогу) и индивиды обретут возможность свободного перемещения, то генетическое разнообразие начнёт вырождаться, что в общем случае приведёт к объединению популяции.

Таблица 3.2. Хеммингово расстояние между хромосомой 0101110

|  |  |
| --- | --- |
| *Хромосомы популяции* | *Количество отличающихся локусов* |
| 0111111 | 2 |
| 0101010 | 1 |
| 0000000 | 4 |
| 0011110 | 2 |
| 1001100 | 3 |
| 1011100 | 4 |
| 1100011 | 4 |
| 1111111 | 3 |

С точки зрения предлагаемой методики, данный подход означает, что при образовании нового поколения макетов пользовательских интерфейсов, используются все предшествующие результаты, т.е. индивиды всех предыдущих поколений. Данный подход неэффективен для исследуемой задачи по следующим причинам:

* использование всей истории вычислений требует больших объёмов памяти;
* обращение к предыдущим поколениям означает рекурсивное увеличение времени поиска кандидатов на скрещивание, что снизит общую скорость работы алгоритма;
* отсутствие функции отбора приведёт к усреднению характеристик формируемого ЧМИ, что воспрепятствует улучшению эргономических характеристик и тем самым не решит поставленную задачу.

Существует математическая модель, описывающая структурные свойства популяции. «F-статистика» строится вокруг двух коэффициентов [40]:

* – коэффициент Панмиксии;
* – коэффициент Фиксации.

Данная модель позволяет предсказывать склонность популяции к панмиксии, через уравнение вида: .

Другой оператор селекции носит название «инбридинг». Особенностью данного способа является вовлечение в процесс скрещивания индивидов с близким генетическим составом. Инбридинг порождает гомозиготность, приводит к проявлению рецессивных генов. В природе это снижает биологическую приспособленность популяции.

Инбридинг широко применяется в процессе селекции: скрещивая индивидов с ярко выраженными целевыми свойствами, можно добиться улучшения целевых характеристик [41]. Как было сказано ранее, в природе инбридинг приводит к формированию потомства с дефектами. Это происходит из-за того, что вероятность образования гомозиготы в таком случае составляет 25%, что приводит к формированию потомства с двумя рецессивными аллелями, которые, свою очередь и несут дефект. Поскольку рецессивные аллели редки в разных популяциях, то вероятность проявления неблагоприятного свойства в потомстве мала. Однако, если в качестве родителей выступают представители одной популяции, то вероятность резко возрастает.

Вопреки частому заблуждению, инбридинг не меняет частоту аллелей, а увеличивает на отношение гомозигот к гетерозиготам; однако, из-за увеличивающейся доли пагубных гомозигот, аллели подвергаются естественному отбору, что на долгом отрезке времени частота снижается именно в инбридинговых популяциях.

С точки зрения рассматриваемой методики, данный метод может быть полезен для корректировки формируемых ЧМИ, вопреки эргономике. Рассмотрим ситуацию, когда идеальный, с точки зрения формального представления, интерфейс не устраивает заказчика. В таком случае, можно детально рассмотреть те поколения ЧМИ, которые более удовлетворяют заказчика, и использовать их чаще, тем самым проводя селекцию.

Аутбридинг признан наилучшим методом селекции [42]. Данный метод заключается в отборе родителей из фенотипически различных популяций, тем самым увеличивая генетическое разнообразие, что снижает уровень наследственных заболеваний. Аутбридинг признан стандартом в животноводческой индустрии. Селекционеры специально избегают вырождения определённых признаков у животных, применяя «свежую кровь». В случае с доминантными признаками, которые могут рано или поздно начать проявляться, прибегают к частичному инбридингу. Стоит отметить, что даже в случае с аутбридингом рецессивные гены остаются в популяции и мигрируют. Впоследствии может быть обнаружена особь, генотип которой содержит много рецессивных генов, представляющих пагубные свойства, которые могут проявиться при последующих скрещиваниях.

С точки зрения ЧМИ, аутбридинг позволяет быстрее прийти к варианту макета с минимальной функцией штрафа, что и требуется по условию задачи. Поэтому данный метод селекции и был выбран в качестве модификации методики.

Непосредственно после выбора родительских индивидов, производится этап рекомбинации для получения нового поколения. Формируемые потомки должны унаследовать свойства обоих родителей. Существует два типа наследования генетической информации: дискретная рекомбинация и кроссинговер.

Дискретная рекомбинация снискала популярность для задач, где хромосомы представлены вещественными генами. Данный вид рекомбинации дополнительно подразделяется на дискретную, линейную и расширенно-линейную. Дискретная рекомбинация применима к генам любого типа:

* двоичные;
* вещественные;
* символьные.

К вещественным генам применяется промежуточная рекомбинация. В начале задаётся числовой диапазон значений генов потомков, в который должны укладываться гены родителей. Общий вид формулы создания потомков следующий:

, где

– потомок,

– родители,

– случайное число на отрезке .

Наилучшее значение для данного алгоритма составляет 0,25. При промежуточной рекомбинации возможно появление генов, которые не встречались ни у одного из родителей. Это позволяет существенно улучшить значение фитнесс-функции потомков в последующих популяциях. Такой метод рекомбинации ещё носит название «Дифференциальное скрещивание».

Таблица 3.3. Вероятность отбора при методе рулетки.

|  |  |  |
| --- | --- | --- |
| *Популяция* | *Приспособленность* | *Вероятность отбора* |
|  | 25 | 0,25 |
|  | 17 | 0,17 |
|  | 34 | 0,34 |
|  | 4 | 0,04 |
|  | 20 | 0,2 |

При линейной рекомбинации множитель выбирается единожды для каждого потомка. Если рассматривать особи популяции как точки в -мерном пространстве, где – количество генов в одной особи, то можно сказать, что при линейной рекомбинации генерируемые точки потомков лежат на прямой, заданной двумя точками – родителями

Алгоритм рекомбинации бинарных генов называют кроссинговером. Существуют следующие виды кроссинговера:

* одноточечный;
* двухточечный.

Модель одноточечного кроссинговера выглядит следующим образом:

– родительские особи,

– хромосомы.

Далее случайным образом определяется точка разрыва , в которой обе хромосомы делятся на две части, после чего происходит обмен генами.

Кроме такого варианта рекомбинации существует также двух и более-точечный кроссинговер. В многоточечном кроссинговере хромосомы принимают вид циклов, которые образуются в результате объединения концов линейной хромосомы. Для совершения операции по замене сегмента одного цикла на сегмент из другого требуется выбор двух точек разреза. В таком случае, одноточечный кроссинговер может быть рассмотрен как кроссинговер с двумя точками, но с одной точкой разреза, зафиксированной в начале строки. Можно заключить, что двухточечный кроссинговер решает задачу аналогичную одноточечному, но в большей мере. При рассмотрении хромосомы в качестве цикла, возможно порождение большого количества стандартных блоков, которые могут совершить «циклический возврат» в конце строки.

Многоточечный кроссинговер позволяет выбрать точек разреза – общее число генов индивида. Выбор точек осуществляется по случайному закону, с запретом повторений. Далее происходит сортировка по возрастанию. При кроссинговере происходит обмен участками хромосом между двух точек, в результате чего образуется два потомка. Участок особи с первым геном до первой точки разреза в обмене не участвует.

Многоточечный кроссинговер обуславливает присутствие нескольких переменных, обозначающих точки разреза.

Все рассмотренные виды кроссинговера так или иначе определяют точки разреза, которые фрагментируют генетический код родительских особей. Выделяют однородный кроссинговер, при котором создаётся маска особи, в каждом локусе которой возможно разделение. Длина маски должна соответствовать количеству геномов особи. Маска задаётся следующим образом.

1. Вводится величина .
2. Выбирается случайное число.
   1. Если случайное число больше , то на -е значение маски принимается за 1.
   2. Иначе – 0.

В результате образуется вектор случайных двоичных чисел. Однородный кроссинговер имеет много общего с многоточечным, однако вектор битовых значений в нём состоит из большего числа элементов. Это гарантирует частое чередование генов родителей в потомках, что, в свою очередь, снижает количество аллелей в генах из-за чего вариативность фенотипических свойств потомков существенно увеличивается. При применении данного вида алгоритма к двоичным строкам, решение сходится к дискретной рекомбинации, как в случае с вещественными хромосомами. В таком случае, процесс называют унифицированным кроссинговером.

Дополнительно рассматривается разновидность триадного кроссинговера. При данной модификации, отбираются не две, а три особи, причём третья выступает в роли маски для двух других. Очевидно, что значение фитнес-функции для третьей особи не играет большой роли, ввиду того что её дальнейшее предназначение не зависит от приспособленности. Затем 10% генов этой особи мутируют. После этого происходит применение маски к генам первого родителя: если гены совпадают, то происходит их передача первому потомку, иначе используются гены второго родителя. Генетический код второго потомка при этом противоположен первому – маска инвертируется.

Во время перестановочного кроссинговера осуществляется произвольный обмен генами между родителями. После этого определяется положение одноточечного кроссинговера и производится перестановка. Затем созданные потомки снова тасуются. Характерной особенностью данного метода выступает то, что изменения прослеживаются не только среди потомства, но и среди родителей. Это позволяет сократить общее число операций в процессе работы алгоритма.

При проведении кроссинговера «с уменьшением замены» вводится специальный оператор, ограничивающий кроссинговер для увеличения вероятности образования принципиально новых особей. Иными словами, снижается количество потомков и идентичным генетическим набором, поскольку они не предоставляют ценности для общего алгоритма. Оператор ограничивает выбор точек разреза таким образом, что они появляются только там, где значения генов различаются.

Алгоритм создаёт мутационного потомка путём случайных изменений генов отдельных родителей. По умолчанию, для неограниченных задач, алгоритм добавляет случайный вектор Гауссова распределения к родителю. Это необходимо для предотвращения преждевременной сходимости алгоритма у локального экстремума.

В общем виде мутация изменяет случайный ген в хромосоме:

По аналогии с кроссинговером, выбор точек для мутации не однозначен: вариативность допускается как в количестве выбранных точек, так и в критериях отбора. Вероятность мутации (обычно ) может быть как фиксированным равновероятным числом в диапазоне , так и функцией, в зависимости от решаемой задачи. Например, вероятность мутации может увеличиваться при крупном скоплении идентичных элементов управления друг за другом. Это можно охарактеризовать как повышение эстетической привлекательности ЧМИ.

Характер мутации напрямую зависит от решаемой задачи. Мутации с фиксированной вероятностью хорошо подходят для унимодальных функций. Самоадаптирующаяся оценка вероятности показывает высокую эффективность на многомодальных функциях.

Существуют следующие критерии остановки генетического алгоритма:

* по числу поколений – алгоритм останавливается, когда создано определённое число поколений;
* по времени – алгоритм прекращает работу через заданное время;
* ограничение фитнес-функции – алгоритм останавливается, когда значение фитнес-функции для лучшей особи становится меньше или равно заданному значению;
* схождение поколений – алгоритм заканчивает работу, когда среднее относительное отклонение фитнес-функции на протяжении указанного числа поколения меньше заданного значения;
* по времени схождения – вычисления прекращаются, когда целевая функция не делает приближения к истинному значению в течение указанного времени;
* по критерию сходимости – критерий сходимости представляет собой либо среднее отклонение, либо геометрическое взвешенное. Для геометрического взвешенного, весовая функция имеет вид , где – количество поколений, предшествующих текущему. Оба критерия сходимости оценивают относительное изменение функции приспособленности в течение заданного количества поколений;
* по величине отклонения – алгоритм выполняется, пока среднее относительное отклонение функции приспособленности на протяжении поколений сходимости меньше заданной величины.

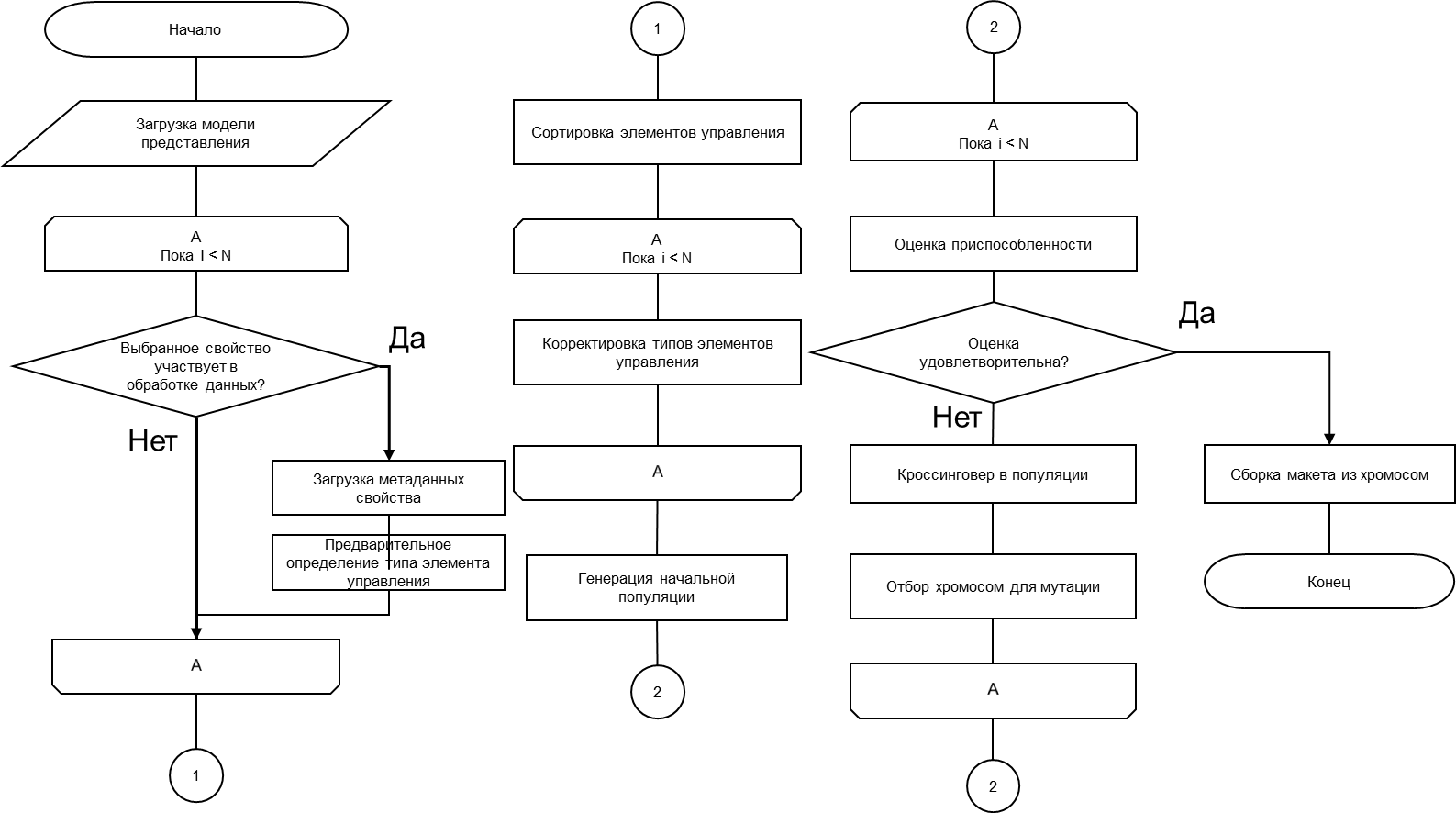


Рис.3.13. Алгоритм формирования макета операторской панели

Выводы по главе III

1. Разработана методика формирования UML-диаграмм для определения команд, доступных оператору.
2. Разработана методика формирования UML-диаграмм последовательности действий для определения порядка следования элементов управления в макете.
3. Разработан алгоритм выбора навигационной парадигмы.
4. Разработан алгоритм выбора элементов управления.
5. Разработан генетический алгоритм компоновки операторской панели с целью минимизации функции штрафа.

# Верификация алгоритма компоновки операторской панели

## Анализ вариантов использования консольных программ \*nix-подобных систем

Для верификации эффективности алгоритма компоновки операторской панели с применением генетического алгоритма была разработана следующая методика. В мире \*nix подобных систем, любая программа или системная утилита имеет основную реализацию в виде программы с интерфейсом командной строки. Это, с одной стороны, позволяет разработчику не отвлекаться от алгоритмической составляющей программы, а с другой стороны, позволяет использовать данное ПО в сценариях автоматизации. Тем не менее, начинающим пользователям, а также пользователям компьютеров без физической клавиатуры трудно использовать эти приложения. Принимая во внимание большое разнообразие графических оболочек на \*nix системах, для многих консольных программ встречается несколько визуальных реализаций. Зачастую каждая графическая интерпретация одной и той же программы построена по своей парадигме и разработана по одной из методик, рассмотренных в главе I. Вследствие этого отдельные группы пользователей предпочитают конкретные варианты такого ПО.

В итоге, алгоритм верификации состоит из следующих этапов.

1. Выбор программы, имеющей как консольный, так и графический интерфейс.
2. Анализ ключей и параметров, передаваемый в консольный интерфейс.
3. Компоновка графического интерфейса для программы по методике описанной в данной диссертации в автоматическом режиме.
4. Оценка функции штрафа, описанной в главе II для уже существующих и вновь разработанного интерфейсов.
5. Сравнение значений функции.

## Программная реализация алгоритма компоновки операторской панели

### Оценка существующих программ на основе графической среды

При разработке программной реализации алгоритма компоновки пользовательского интерфейса возникла задача выбора подходящей графической среды. Главной особенностью искомого решения была возможность отображения интерфейса в графическом виде и параллельная работа функции вычисления штрафа. Для этого требовалось, чтобы была предусмотрена возможность расстановки элементов управления в абсолютных координатах, без каких-либо технологий плавающих разметок. Кроме того, было необходимо разработать средство оценки уже существующих программ, без доступа к их исходным кодам.

В результате был сделан выбор в пользу графической среды GTK+ 3. GTK+ 3 содержит встроенный визуальный отладчик GtkInspector. GtkInspector внедряется в процесс приложения. Он позволяет выводить подробную отладочную информацию касательно графического интерфейса, такую как:

* иерархия элементов управления;
* идентификаторы окон;
* свойства элементов управления
* стили, применённые к конкретному элементу.

Данная программа также позволяет манипулировать свойствами макета, но это не востребовано в рамках данного исследования.

В процессе верификации, для каждого из исследуемых приложений рассчитывалась функция штрафа, приведённая в главе II c помощью обхода элементов их интерфейса описанным выше отладчиком. Далее эти сведения использовались для сравнения с собственной методикой.

Рис.4.1. Значение сравнение функции штрафа для существующих реализаций

### Разработка верификатора компоновки ЧМИ с применением паттерна MVVM

Разработка программы верификации генетического алгоритма ЧМИ проводилась в интегрированной среде разработки «Microsoft Visual Studio 2015: Community» на языке C#. Приложение разрабатывалось с применением паттерна Model-View-ViewModel (MVVM), поскольку данный подход наилучшим образом соответствует поставленной задаче: единожды сформировав модель данных, многократно экспериментировать с различными вариантами представления. Для сохранения результатов применялась система Object-relation-mapper (ORM) Entity Framework в связке с СУБД «Microsoft SQL Server Express 2016: Local DB».

Программа состоит из трёх уровней:

* Модели данных
* Бизнес-логика
* Представление

Модели данных представлены структурами, с которыми оперирует программа. В них входят формализованное представление UML диаграмм и модель индивида для генетического алгоритма.

Представление выражено макетами графического интерфейса на языке разметки XAML. Язык разметки XAML позволяет довольно гибко описать свойства объектов и их композицию. В данной работе XAML используется не только для описания интерфейса верификатора, но и для отображения промежуточных результатов ГА в ходе оптимизации макета.

Наиболее значимая часть программы представляет собой бизнес-логика. Здесь описываются составные части алгоритма, такие как функции селекции, скрещивания, мутации и оценки, согласно математической модели, описанной во второй главе данной работы.

### Ускорение процесса компоновки с помощью параллельных SIMD вычислений

Рассматриваемая задача относится к категории ограниченных ресурсами процессора. Это означает, что скорость получения результата зависит не от пропускной способности ввода-вывода, а от количества операций в единицу времени. Такие задачи можно вычислять параллельно, по принципу SIMD (single instruction, multiple data – одиночный поток команд, множественный поток данных) Генетические алгоритмы также поддерживают распараллеливание. Это достигается путём параллельной обработки различных популяций. В определённые моменты времени, происходит сравнение значений функции приспособленности у популяций в разных потоках. На базе вычисленных значений отбирается количество популяций с наилучшей приспособленностью в соответствии с количеством параллельных потоков. При этом популяции с неудовлетворительным значением фитнес-функции уничтожаются.

При решении данной задачи для обеспечения параллелизма использовалась программно-аппаратная платформа CUDA. Данная платформа позволяет задействовать ресурсы графического адаптера для многопоточного выполнения алгоритма. В программной модели CUDA, группа блоков нитей, выполняющих одно ядро, носит название сетки (Grid) [43]. При использовании динамического параллелизма параллельная сетка осуществляет запуск ядер, называемых дочерними сетками. Дочерняя сетка наследует от родительской определённые атрибуты и ограничения, такие как объём L1 кэша, конфигурацию разделяемой памяти и размер стека. В частности, если родительская сетка содержит 128 блоков по 64 нити, и процедура запуска дочерних ядер не регулируется, то в общей сложности производится 8192 запусков ядер. Если при этом требуется только один запуск дочерней сетки на поток, то каждая нить каждого блока должна осуществлять запуск ядра индивидуально. Сетки, запущенные с применением динамического параллелизма, образуют строгую иерархию. Это означает, что дочерняя сетка всегда завершает работу раньше, чем родительская, при отсутствии явной синхронизации.

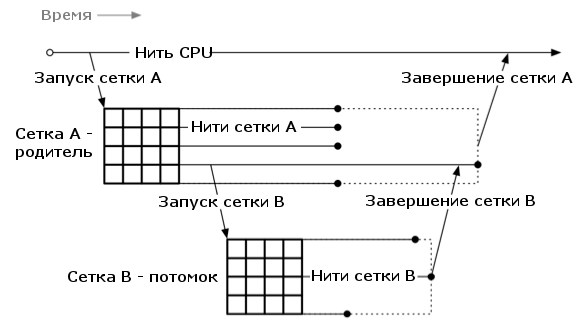


Рис.4.2. Иерархия динамического параллелизма

Если же родительскому ядру требуются результаты вычислений дочернего, то необходимо явно произвести синхронизацию контекста. Эта функция ожидает завершения всех дочерних сеток, запущенных блоком нитей, вызвавшим её. Из-за вложенности, эта функция также ожидает завершения всей иерархии вызова и сообщает об ошибках, при их возникновении. Стоит учитывать, что нить, инициировавшая процесс синхронизации не в состоянии определить какие из дочерних ядер, уже завершили свою работу. Тем не менее, если требуется синхронизация на уровне блока, то необходимо предварительно опросить каждую из дочерних сеток перед общей синхронизацией. После этого производится синхронизация нитей, для обеспечения дальнейших вычислений. В общем случае, описываемый процесс довольно затратен по ресурсам, поскольку он может спровоцировать приостановку выполнения блока и освобождение оперативной памяти. В частности, не стоит использовать данный метод в конце функции ядра, так как в данном случае всегда осуществляется синхронизация на системном уровне.

Родительская сетка всегда зависит от данных, которые записывают и извлекают дочерние сетки в глобальную память. Для работы в таком режиме среда исполнения CUDA гарантирует единообразие представления глобальной памяти для сеток всех уровней в моменты запуска и остановки уровней. Это означает, что если родитель осуществляет запись по определённому адресу, а затем запускает дочернюю сетку, то потомок обязательно получит значение, записанное родителем. Это правило работает и наоборот – если потом совершает запись в память, а родитель осуществляет синхронизацию, то родитель гарантированно получит значение, записанное потомком. Это также работает в случае последовательно исполнения нескольких потомков, когда слои более низкого уровня получают доступ к сведениям, записанным элементами предыдущих уровней без принудительной синхронизации.

Однако представление памяти не едино в момент выполнения конструктора ядра. Во избежание состояния гонок не следует осуществлять запись в память, которая может быть прочитана потомком, в период между запуском ядра и принудительной синхронизацией.

Таблица ниже перечисляет ограничения на типы указателей, передаваемых в дочерние ядра

Таблица 4.1. Допустимые для передачи в GPU типы указателей

|  |  |
| --- | --- |
| *Могут передаваться* | *Не могут передаваться* |
| Глобальная память | Разделяемая память |
| Память хоста | Локальная память |
| Константная память |  |

Результат разыменования указателя в потомке, передача которого запрещена приводит к неопределённому состоянию. В действительности, попытка скомпилировать код, содержащий запрещённую операцию, приведёт к ошибке компиляции. Однако это можно обойти, обернув указатель в структуру данных. Тем не менее такой код не будет работать как задумано. Указатель на локальную память может ссылаться только на те участки памяти, которые доступны из дочерних сеток. Начиная с CUDA версии 5.5 ошибки такого типа не перехватываются анализаторами памяти.

Стоит также учитывать, что передача указателя на локальную переменную довольно часто используется в обычных CPU программах, но запрещена в динамическом параллелизме. Несмотря на эффективность данного приёма в CPU, для GPU требуется иной подход. Передача указателя на глобальную функцию допустима, но не так эффективна ввиду большого числа дочерних сеток и малого количества глобальных переменных. В перечень наиболее часто встречающихся приёмов входят:

* использование функции *malloc* на устройстве: данный способ довольно медленный из-за немасштабируемости механизма выделения памяти;
* использование предопределённых структур: для случаев, когда количество образующихся дочерних сеток предсказуемо, появляется возможность предварительно выделить память по одной структуре на сетку;
* разработка собственного механизма выделения памяти: затратно по усилиям и может привести к ограниченности полученного решения, однако в случае с недетерминированным количеством запускаемых сеток является единственным методом;
* использование сторонних механизмов выделения памяти: несмотря на открытый доступ к некоторым из них, их качество не удовлетворяет производственные нужды и может вызвать серьёзные проблемы в коммерческих программах.

По умолчанию сетки, запущенные в блоке нитей, исполняется последовательно: следующая сетка запускается только по завершении предыдущей. Так происходит даже если сетки запускаются различными нитями внутри одного блока. Зачастую требуется больше параллелизма. Это возможно с помощью механизма потоков в среде CUDA.

Все потоки, создаваемые в устройстве, являются неблокирующими, что подразумевает, что они не поддерживают явную синхронизацию с помощью стандартного *NULL-*потока. Ядра, запускаемые в других потоках, могут выполняться одновременно. Потоки, создаваемые в других блоках нитей, включая стандартный *NULL*-поток, считаются другими. Однако даже при этом параллельное исполнение не гарантируется: такое решение разработано в первую очередь для более равномерного распределения ресурсов GPU между задачами. Программы, зависящие от параллельного исполнения определённых сеток или блоков нитей некорректны.

Будучи созданными, поток на устройстве может быть использован любой нитью в пределах того же блока, что и поток. Однако по завершении работы блока, поток не может быть использован другими блоками хоста. Соответственно и потоки, созданные на хосте, не могут быть задействованы на устройстве.

В настоящее время в среде исполнения CUDA отсутствуют механизмы синхронизации потоков. Единственная возможность синхронизации осуществляется путём ожидания завершения всей работы, назначенной потоку.

На хосте потоки можно уничтожать. Если в момент уничтожения поток всё ещё выполняет работу, то ресурсы будут высвобождены по завершении работы. Таким образом отпадает необходимость ожидания завершения работы потоков для высвобождения ресурсов.

Несмотря на существование системы событий на GPU, единственное официальное применение этой системы состоит в передаче команд потокам. Использование событий для выполнения действий по расписанию или для синхронизации по событиям в настоящий момент не поддерживается.

Программы, выполняющиеся рекурсивно, обычно характеризуются глубиной рекурсии. В динамическом параллелизме существуют две концепции реализации таких программ:

* глубина вложенности – задаёт наибольшую глубину вложенности сеток, отсчёт которых начинается с хоста;
* глубина синхронизации – задаёт наибольший уровень, с которого вызывается синхронизация.

Обычно глубина синхронизации не превосходит глубину вложенности. Если явная синхронизация выполняется не на каждом уровне, то глубина синхронизации сильно меньше.

На каждом уровне, не превышающем максимальную глубину синхронизации, возможна замена родительского блока на дочернюю сетку, из-за чего данные родительской сетки перемещаются в хранилище. На GPU класса GK110 (Geforce GTC 780 Ti, Tesla K20 и т.д) каждый уровень может резервировать до 150Мб памяти, в зависимости от максимального количества нитей, доступных на устройстве (возможны вариации из-за количества памяти на устройстве). При этом всегда резервируется несколько больше памяти, чем фактически необходимо.

В CUDA существуют специальные функции для оценки максимальной глубины синхронизации на устройстве. Если запросить явную синхронизацию на уровне, превышающем максимальную глубину, то будет выдано сообщение об ошибке, однако дочерние ядра всё равно будут запущены несмотря на то, что результат их работы будет вероятнее всего ошибочным ввиду пропущенного этапа синхронизации.

По умолчанию память резервируется на два уровня синхронизации. В среде исполнения CUDA максимальная глубина синхронизации отображает количество уровней, для которых резервируется хранилище. Причём хост учитывается как дополнительный уровень, поэтому фактическое значение на единицу меньше отображаемого.

Дополнительно существует аппаратное ограничение на максимальную глубину вложенности. В стандарте Compute Capability 3.5, предел составляет 24 уровня.

Ещё одним ограничением выступает число ожидающих дочерних сеток, которые могут выполняться, быть приостановленными или находиться в очереди на запуск. Буфер ожидания запуска представляет собой структуру, используемую для поддержания очерёдности запуска и отслеживания выполняющихся ядер. Если происходит запуск ядра при заполненном буфере, то поведение системы зависит то версии CUDA. В CUDA 5 запускаемая сетка будет сброшена и не будет запущена. Последующие обращения к функции возврата ошибки будут сообщать соответствующее значение. По умолчанию буфер рассчитан на 2048 ожидающих дочерних ячеек. Размер буфера можно скорректировать в пределах аппаратных ограничений устройства.

В CUDA 6 и более поздних версиях был введён динамический виртуализируемый пул, в добавок к ранее существующему буферу фиксированного объёма. В таком случае, предел глубины относится к фиксированному буферу. Среда исполнения в начале пытается поместить вновь созданную пул фиксированного размера и в случае неудачи использует виртуальный. Несмотря на очевидное преимущество в виде возможности запуска большего числа сеток, расходы, сопряжённые с использованием виртуального пула довольно высоки. Ниже приведен график зависимости скорости работы функции в зависимости от первоначально заданного фиксированного пула.

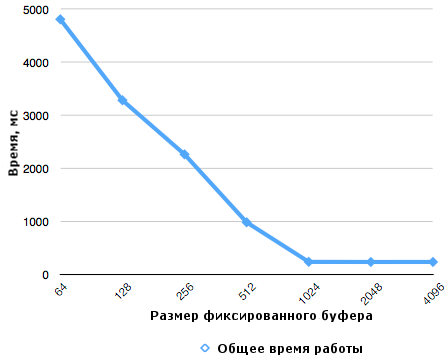


Рис.4.3. Зависимость скорости работы функции от размера фиксированного пула

Как видно на графике, использование виртуального пула замедляет работу программы более, чем в 20 раз. В примере имеет место явное занижение размера фиксированного пула, однако это наглядно отражает суть проблемы. Чем более точно можно предусмотреть количество сеток, тем лучше производительность программы.

### Минимаксная оценка скорости сходимости генетического алгоритма в зависимости от алгоритмов селекции и скрещивания

Учитывая тот факт, что работа генетического алгоритма включает в себя случайный фактор, невозможно строго предугадать количество итераций до достижения желаемого значения. Тем не менее, в задачи данной диссертации входит выявление наилучшего варианта алгоритма, позволяющего минимизировать функцию штрафа за минимальное количество шагов. В таком случае следует воспользоваться минимаксным критерием оценки эффективности алгоритмов.

Критерий формулируется следующим образом:

1. Вначале формируется матрица стратегий. Столбцы соответствуют возможным вариантам запуска. В ячейках фиксируется ожидаемое количество итераций для достижения минимума функции штрафа для данного исхода при выбранном методе.
2. Затем формируется матрица сожаления. В ячейках матрицы указывается величина сожаления – различие между максимальным результатом при выбранном исходе (максимальное количество итераций в рассматриваемом столбце) и результатом при выбранной методике
3. Минимаксное решение позволяет определить стратегию, при которой максимальное сожаление минимально. Для этого сначала для каждой стратегии (методики) выбирается максимальная величина сожаления, а потом выбирается та строка (решение), максимальное сожаление которого минимально.

Таблица 4.2. Пример матрицы сожаления для ГА

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | *Одноточечный* | *Двуточечный* | *Триадный* | *Min* |
| Аутбридинг | 10658 | 13284 | 25684 | 10658 |
| Инбридинг | 12932 | 12651 | 14945 | 12651 |
| Панмиксия | 15987 | 16245 | 20852 | 15987 |

### Визуализация результатов средствами MATLAB

Для наглядного представления результатов использовался пакет MATLAB. Данные, полученные из программы, были выведены в формате CSV (Comma-separated values). В таком формате их можно просматривать практически в любой программе для работы с табличными данными (например, Microsoft Excel, или Power BI). В данной работе, результаты исследования загружались в программу MATLAB. Далее производилась нормализация и визуализация сведений.

MATLAB позволяет отображать двумерные массивы. В частности, если имеется векторы и , то возможно их отображение в качестве массива точек на плоскости с координатами . По умолчанию MATLAB осуществляет линейную интерполяцию заданных точек полиномом нулевой степени, т.е. образует ломанную линию.

Определённые трудности возникают при визуализации функции от многих переменных. Для визуализации функции от двух переменных было проделано следующее.

1. Предположим, что требуется отобразить функцию . Сначала необходимо сформировать матрицы и , представляющие собой повторяющиеся строки и столбцы в пределах области определения функции. Эти матрицы будут использоваться в дальнейшем для вычисления значения функции в точках пересечения.
2. С помощью встроенной функции *meshgrid* область определения функции, заданной двумя векторами и преобразуется в матрицы соответственно.

Выводы по главе IV

1. Произведён анализ вариантов использования консольных программ \*nix-подобных систем.
2. Разработана программная реализация алгоритма компоновки операторской панели, включающая в себя:
   * оценку существующих программ на основе графической среды;
   * разработку верификатора компоновки ЧМИ с применением паттерна MVVM;
   * ускорение процесса компоновки с помощью параллельных SIMD вычислений;
   * минимаксную оценка скорости сходимости генетического алгоритма в зависимости от алгоритмов селекции и скрещивания;
   * визуализацию результатов средствами MATLAB.

Заключение

В ходе выполнения диссертационного исследования получены следующие результаты.

1. Выявлены характерные недостатки современных методов проектирования ЧМИ, основным из которых выступает отсутствие адаптивности под системы с мультимодальным вводом.
2. Исследованы современные эргономические требования к промышленному оборудованию и актуальные модели человеческого восприятия. Формализованы основные моменты Гештальтпсихологии.
3. Разработана математическая модель эргономических качеств операторской панели. Минимизируя значение данной функции можно итеративно повышать удобство использования операторской панели.
4. Исследованы математический аппарат минимизации недифференциируемых функции. Рассмотрены вариации Генетических Алгоритмов, алгоритмов селекции и скрещивания. Предложен ряд модификаций для соответствия предметной области.
5. Разработана программная реализация предложенных алгоритмов и проведена минимаксная оценка полученных результатов. Произведено сравнение с уже существующими ЧМИ. Математически доказано улучшение эргономических свойств на ~5%.
6. Предложены варианты повышения производительности предложенного алгоритма путём параллелизации с применением графических ускорителей.

Список литературы

x

|  |  |
| --- | --- |
| 1. | Eva Flaspöler A.H.P.P.A.D.R.B.G.T.B.N.H.S.K.A.S.A.A.W. The human machine interface as an emerging risk. European Agency for Safety and Health at Work , 2010. |
| 2. | John B.E., Kieras D.E. The GOMS Family of User Interface Analysis Techniques: Comparison and Contrast. 1996. |
| 3. | Грэхэм Л. Естествознание, философия и науки о человеческом поведении в Советском Союзе. |
| 4. | 13407:1999 I. Human-centred design processes for interactive systems. |
| 5. | Himma K.E., Tavani H.T. The Handbook of Information and Computer Ethics. 2008. |
| 6. | Electronics B. Trends in HMI [Электронный ресурс] // Automation.com: [сайт]. [2010]. URL: https:/​/​www.automation.com/​library/​white-papers/​trends-in-hmi |
| 7. | IoT - Internet of Things [Электронный ресурс] // Oracle: [сайт]. [2017]. URL: https:/​/​www.oracle.com/​solutions/​internet-of-things/ |
| 8. | Schmidt M.B. Mobile UI ergonomics: How hard is it really to tap different areas of your phone interface? // Medium. Октябрь 2014. |
| 9. | Over-the-Air (OTA) [Электронный ресурс] // Technology - Gemalto: [сайт]. [2010]. URL: https:/​/​www.gemalto.com/​companyinfo/​digital-security/​techno/​ota |
| 10. | Instrumenting Your App for Telemetry and Analytics [Электронный ресурс] // Windows Developer BlogWindows Developer Blog: [сайт]. [2014]. URL: https:/​/​blogs.windows.com/​buildingapps/​2014/​03/​20/​instrumenting-your-app-for-telemetry-and-analytics/ |
| 11. | Maggio. Guide to software usability principles. Progea S.r.l., 2011. |
| 12. | ГОСТ Р ИСО 9241-210–2012. Эргономика взаимодействия человек–система. Часть 210. Человеко-ориентированное проектирование интерактивных систем. Москва: Стандартинформ, 2013. |
| 13. | EN ISO 92412010:2010 Ergonomics of human-system interaction – Part 210: Human-centered design for interactive systems. |
| 14. | Miller G.A. The magical number seven, plus or minus two: some limits on our capacity for processing information // The Psychological Review, Vol. 63, 1956. pp. 81-97. |
| 15. | Hick’s Law: Making the choice easier for users [Электронный ресурс] // Interaction Design Foundation: [сайт]. [2017]. URL: https:/​/​www.interaction-design.org/​literature/​article/​hick-s-law-making-the-choice-easier-for-users |
| 16. | Fitts P.M. The information capacity of the human motor system in controlling the amplitude of movement // Journal of Experimental Psychology, Vol. 47, No. 6, 1954. pp. 381-391. |
| 17. | MacKenzie I.S. B.W. Extending Fitts' law to two dimensional tasks // Proceedings of the CHI '92 Conference on Human Factors in Computing Systems. New York. 1992. |
| 18. | Imprinting | learning behaviour [Электронный ресурс] // Britannica.com: [сайт]. [2015]. URL: https:/​/​www.britannica.com/​topic/​imprinting-learning-behaviour |
| 19. | Gruhn P. Human machine interface: the good, the bad, and the ugly (and what makes them so) // Proceedings of the 66th Annual Instrumentation Symposium for the Process Industries. Research Triangle Park, N.C. 2011. |
| 20. | Bill Hollifield D.O.I.N.E.H. The high performance HMI handbook: a comprehensive guide to designing, implementing and maintaining effective HMIs for industrial plant operations. Houston: TX : Plant Automation Services, Inc, 2008. |
| 21. | Mica Endsley B.B.D.J. Designing for situation awareness: an approach to user-centered design. Boca Raton: FL : CRC Press, 2003. |
| 22. | Директива Европейского Парламента и Совета Европейского Союза 2006/42/EC о машинах и механизмах. 2006. |
| 23. | Fraser I. Guide to application of the Machinery Directive 2006/42/EC. Brussels: European Commission, Enterprise and industry, 2010. |
| 24. | EN 13861:2011, Safety of machinery – Guidance for the application of ergonomics standards in the design of machinery. Brussels: CEN, 2011. |
| 25. | EN 614 (части 1-2) Safety of machinery – Ergonomic design principles. Brussels : CEN. |
| 26. | EN 894 (parts 1-4) Safety of machinery – Ergonomics requirements for the design of displays and control actuators. Brussels : CEN. |
| 27. | BS EN ISO 10075-1:2000 Ergonomic principles related to mental workload. General terms and definitions. |
| 28. | ISO 10075-2 Ergonomic principles related to mental workload -- Part 2: Design principles. |
| 29. | DIN EN 614-1-2009. Безопасность машин и механизмов. Эргономические принципы проектирования. Часть 1. Терминология и общие принципы. 2009. |
| 30. | EN 894-3:2000. Эргономические требования к оформлению индикаторов и органов управления. 2000. |
| 31. | Директива N 90/270/ЕЭС. О минимальных требованиях безопасности при работе с дисплейным оборудованием. |
| 32. | ГОСТ Р ИСО 9241-110-2016. Эргономика взаимодействия человек-система. Часть 110. Принципы организации диалога. |
| 33. | Back R.-J. P.L..P.P.I. Second International Conference on the Uniﬁed Modeling Language // Analyzing UML Use Case as Contracts. Springer Verlag. 1999. |
| 34. | Shen W. L.S. ICFEM 2003 International Conference on Formal Engineering Methods // Formalization, Testing and Execution of a Use Case Diagram. Singapore. 2003. |
| 35. | BarrettS. S... 3rd International Symposium on Theoretical Aspects of Software Engineering // Merging of Use Case Models: Semantic Foundations. IEEE. 2009. |
| 36. | Blankenship E. What is a UI Control? [Электронный ресурс] URL: https:/​/​experience.sap.com/​basics/​what-is-a-ui-control/ |
| 37. | Genetic Algorithm [Электронный ресурс] // MATLAB & Simulink: [сайт]. [2016]. URL: https:/​/​www.mathworks.com/​discovery/​genetic-algorithm.html |
| 38. | Панченко Т.В. Генетические алгоритмы. Астрахань: Издательский дом «Астраханский университет» , 2007. |
| 39. | Genetic panmixia and demographic dependenc [Электронный ресурс] // PMV: [сайт]. URL: https:/​/​www.ncbi.nlm.nih.gov/​pmc/​articles/​PMC3183912/ |
| 40. | Understanding Analysis of Variance (ANOVA) and the F-test [Электронный ресурс] // The Minitab Blog: [сайт]. [2016]. URL: http:/​/​blog.minitab.com/​blog/​adventures-in-statistics-2/​understanding-analysis-of-variance-anova-and-the-f-test |
| 41. | Inbreeding - an overview [Электронный ресурс] // ScienceDirect Topics: [сайт]. URL: https:/​/​www.sciencedirect.com/​topics/​biochemistry-genetics-and-molecular-biology/​inbreeding |
| 42. | Christopher A. Monson K.C.S. nbreeding Depression and Outbreeding Depression Are Evident in Wild-Type Zebrafish Lines, 2010. |
| 43. | Adinets A. CUDA Dynamic Parallelism API and Principles [Электронный ресурс] [2014]. URL: https:/​/​devblogs.nvidia.com/​cuda-dynamic-parallelism-api-principles/ |
| 44. | Kieras D..J.B. The GOMS Family of Analysis Techniques: Tools for Design and Evaluation. 1994. |
| 45. | Kantowitz B.H.&.S.R.D. Human factors: Understanding people-system relationships. Нью-Йорк: Wiley, 1983. |
| 46. | Karwowski W. The disciplines of human factors and ergonomics // In: Handbook of human factors and ergonomics. Hoboken: Wiley, 2012. pp. 3-37. |
| 47. | Oborne D.J. Ergonomics at work. Chichester: Wiley, 1996. |
| 48. | ISO 6385:2004 Ergonomic principles in the design of work systems. |
| 49. | Sanders M.S.&.M.E.J. Human factors in engineering and design. New York: McGraw Hill, 1993. |
| 50. | ISO 6385:2004 Ergonomic principles in the design of work systems. |
| 51. | Hendrick H.W.&.K.B.M. Macroergonomics: An introduction to work systems design. Santa Monica: Human Factors and Ergonomics Society, 2001. |
| 52. | Miller C..N.P..D.N.F..M.B..N.M..P.R.&.W.I. Strasbourg: Indigo, THESEUS Cluster 2: Psychology and Human-Machine Systems – Report. pp. 22-38. |
| 53. | Kantowitz B.H.&.S.R.D. Human factors: Understanding people-system relationships. Нью-Йорк: Wiley, 1983. |
| 54. | Hacker W. Mental workload // In: Encyclopaedia of Occupational Health and Safety (vol. 1). Geneva: ILO, 1998. pp. 41-43. |
| 55. | Nachreiner F. Ergonomics and standardization // In: Encyclopaedia of Occupational Health and Safety. Geneva: ILO, 1998. pp. 11-14. |
| 56. | ISO 6385:2004 Ergonomic principles in the design of work systems. |
| 57. | Kirchner J.H..B.E. Ergonomie für Konstrukteure und Arbeitsgestalter. REFA-Fachbuchreihe Betriebsorganisation. Munich: Carl-Hanser-Verlag, 1990. |
| 58. | EN ISO 13857: Safety of machinery – Safety distances to prevent hazard zones being reached by upper and lower limbs. Brussels : CEN. |
| 59. | EN 1005 series (parts 1-5). Safety of machinery – Human physical performance. Brussels : CEN. |
| 60. | Enns J.T. Gestalt Principles of Perception // In: Encyclopedia of Cognitive Science. London: Nature Publishing Group, 2003. |
| 61. | Zühlke D. Nutzergerechte Entwicklung von Mensch-Maschine-Systemen. Berlin: Springer, 2012. |
| 62. | Goldstein E.B. Sensation and Perception. USA: Wadsworth, 2010. |
| 63. | Richter P. Mental fatigue // In: Encyclopaedia of Occupational Health and Safety. Geneva : ILO, 1998. pp. 46-47. |
| 64. | ISO 10075 series (parts 1-3). Ergonomic principles related to mental workload. Geneva: ISO. |
| 65. | Nachreiner F..N.P.&.M.I. Safety Science, 2006. pp. 5-26. |
| 66. | Sanders M.S.&.M.E.J. Human factors in engineering and design. New York: McGraw Hill, 1993. |

x

Приложение

Приложение 1. Программная реализация модифицированного алгоритма выбора типа элемента управления.

public class ResourceTemplateSelector : DataTemplateSelector

{

public static bool GetIsInput(DependencyObject obj)

{

return (bool)obj.GetValue(IsInputProperty);

}

public static void SetIsInput(DependencyObject obj, bool value)

{

obj.SetValue(IsInputProperty, value);

}

public static readonly DependencyProperty IsInputProperty =

DependencyProperty.RegisterAttached("IsInput", typeof(bool), typeof(ResourceTemplateSelector), new FrameworkPropertyMetadata(false, FrameworkPropertyMetadataOptions.Inherits));

public override DataTemplate SelectTemplate(object item, DependencyObject container)

{

Type itemType = null;

var res = item as ResourceBlockModel;

if (res != null)

itemType = res.Type;

else

{

var prop = item as PropertyBlockModel;

if (prop != null)

itemType = prop.Type;

}

if (itemType != null)

{

Type viewType = null;

if (GetIsInput(container))

{

if (itemType.GetInterface(nameof(ICollection)) != null)

viewType = typeof(CollectionInputView); // Если подали список

else

{

if (itemType.IsEnum)

viewType = typeof(EnumView);

else

{

switch (itemType.Name)

{

case nameof(Int32):

case nameof(Double):

case nameof(String):

case nameof(TimeSpan):

// Значения, которые могут быть представлены текстом

viewType = typeof(TextInputView);

break;

case nameof(DateTime):

viewType = typeof(DateTimeView);

break;

default:

viewType = typeof(ResourceView);

break;

}

}

}

}

else

{

if (itemType.GetInterface(nameof(ICollection)) != null)

viewType = typeof(CollectionView); // Если подали список

else

{

switch (itemType.Name)

{

case nameof(Int32):

case nameof(Double):

case nameof(String):

case nameof(TimeSpan):

// Значения, которые могут быть представлены текстом

viewType = typeof(Properties.TextView);

break;

case nameof(ToolData):

viewType = typeof(ToolDataOutputView);

break;

default:

viewType = typeof(DefaultView);

break;

}

}

}

if (viewType != null)

{

var factory = new FrameworkElementFactory(viewType);

var dataTemplate = new DataTemplate(itemType);

dataTemplate.VisualTree = factory;

return dataTemplate;

}

}

return null;

}

}

Приложение 2. Фрагмент инициализации параллельного исполнения ГА

|  |
| --- |
|  |
|  | #include <string> |
|  | #include <ctime> |
|  | #include <cstring> |
|  |  |
|  | #include "common.h" |
|  | #include "log.h" |
|  | #include "ga\_cpu.h" |
|  | #include "ga\_gpu.h" |
|  |  |
|  | using namespace std; |
|  |  |
|  | int main(int argc, const char \* argv[]) { |
|  | // Журналирование |
|  | Logger gen\_log; |
|  |  |
|  | // Параметры ГА |
|  | float prob\_mutation = 0.15f; // Вероятность мутации |
|  | float prob\_crossover = 0.8f; // Вероятность кроссинговера |
|  | int ui\_seed = 12345678; // Начальная популяция |
|  | int ga\_seed = 87654321; |
|  |  |
|  | // Параметры макета |
|  | const int ui\_width = 10000; // Ширина экрана |
|  | const int ui\_height = 10000; // Высота экрана |
|  |  |
|  | // The test cases |
|  | const int iterations = 1; // Количество полных циклов |
|  | const int num\_cases = 1; // Количество проверок |
|  | int cases[num\_cases][3] = // кол-во элементов, размер популяции, макс. кол-во |
|  | { |
|  | // {25, 100, 1000}, |
|  | // {25, 1000, 1000}, |
|  | // {25, 10000, 100}, |
|  | {25, 100000, 10} |
|  | // {50, 100, 1000}, |
|  | // {50, 1000, 1000}, |
|  | // {50, 10000, 100}, |
|  | // {50, 100000, 10}, |
|  | // |
|  | // {100, 100, 1000}, |
|  | // {100, 1000, 1000}, |
|  | // {100, 10000, 100}, |
|  | // {100, 100000, 10}, |
|  | // |
|  | // {250, 100, 1000}, |
|  | // {250, 1000, 1000}, |
|  | // {250, 10000, 100}, |
|  | // {250, 100000, 10} |
|  | }; |
|  |  |
|  | // Время |
|  | clock\_t iter\_time, total\_time; |
|  |  |
|  | // Путь к результату |
|  | std::string path = "./"; |
|  |  |
|  | // Перебор всех элементов |
|  | for (int i=0; i<num\_cases; i++) |
|  | { |
|  | // Параметры ГА |
|  | int num\_elements = cases[i][0]; |
|  | int pop\_size = cases[i][1]; |
|  | int max\_gen = cases[i][2]; |
|  |  |
|  | // Формирование строк |
|  | std::string c\_timing\_path; |
|  | std::string c\_gen\_path; |
|  | std::string c\_stats\_path; |
|  | std::string g\_timing\_path; |
|  | std::string g\_gen\_path; |
|  | std::string g\_stats\_path; |
|  |  |
|  | std::string c0, c1, c2; |
|  | std::string g0, g1, g2; |
|  |  |
|  | c0 = to\_string(num\_elements) + "\_" + to\_string(pop\_size) + "-cpu\_gen.csv"; |
|  | c1 = to\_string(num\_elements) + "\_" + to\_string(pop\_size) + "-cpu\_timing.csv"; |
|  | c2 = to\_string(num\_elements) + "\_" + to\_string(pop\_size) + "-cpu\_stats.csv"; |
|  | g0 = to\_string(num\_elements) + "\_" + to\_string(pop\_size) + "-gpu\_gen.csv"; |
|  | g1 = to\_string(num\_elements) + "\_" + to\_string(pop\_size) + "-gpu\_timing.csv"; |
|  | g2 = to\_string(num\_elements) + "\_" + to\_string(pop\_size) + "-gpu\_stats.csv"; |
|  |  |
|  | // Компоновка интерфейса |
|  | Ui ui(num\_elements, ui\_height, ui\_width, ui\_seed); |
|  |  |
|  | // Строки журналирования |
|  | c\_timing\_path = path + c0; |
|  | c\_gen\_path = path + c1; |
|  | c\_stats\_path = path + c2; |
|  | g\_timing\_path = path + g0; |
|  | g\_gen\_path = path + g1; |
|  | g\_stats\_path = path + g2; |
|  |  |
|  | cout << endl; |
|  | cout << "###############################################################################" << endl; |
|  | cout << "##### CPU - START" << endl; |
|  | cout << "###############################################################################" << endl << endl; |
|  |  |
|  | // Время CPU |
|  | gen\_log.start(c\_gen\_path, c\_timing\_path, c\_stats\_path); |
|  | total\_time = clock(); |
|  | for (int j=0; j<iterations; j++) |
|  | { |
|  | iter\_time = clock(); |
|  | execute(pop\_size, max\_gen, prob\_mutation, prob\_crossover, ui, gen\_log, ga\_seed); |
|  | gen\_log.write\_stats(j + 1, "CPU", end\_clock(iter\_time), |
|  | prob\_mutation, prob\_crossover, pop\_size, max\_gen, ui\_seed, |
|  | ga\_seed, ui\_width, ui\_height, num\_elements); |
|  | } |
|  | gen\_log.write\_stats(-1, "CPU", end\_clock(total\_time), |
|  | prob\_mutation, prob\_crossover, pop\_size, max\_gen, ui\_seed, |
|  | ga\_seed, ui\_width, ui\_height, num\_elements); |
|  | gen\_log.end(); |
|  |  |
|  | cout << endl; |
|  | cout << "###############################################################################" << endl; |
|  | cout << "CPU - END" << endl; |
|  | cout << "###############################################################################" << endl << endl; |
|  |  |
|  | cout << "===============================================================================" << endl << endl; |
|  |  |
|  | cout << "###############################################################################" << endl; |
|  | cout << "GPU - START" << endl; |
|  | cout << "###############################################################################" << endl << endl; |
|  |  |
|  |  |
|  | gen\_log.start(g\_gen\_path, g\_timing\_path, g\_stats\_path); |
|  | g\_execute(pop\_size, 1, prob\_mutation, prob\_crossover, ui, gen\_log, ga\_seed); |
|  | gen\_log.end(); |
|  |  |
|  | // GPU timing |
|  | gen\_log.start(g\_gen\_path, g\_timing\_path, g\_stats\_path); |
|  | total\_time = clock(); |
|  | for (int j=0; j<iterations; j++) |
|  | { |
|  | iter\_time = clock(); |
|  | g\_execute(pop\_size, max\_gen, prob\_mutation, prob\_crossover, ui, gen\_log, ga\_seed); |
|  | gen\_log.write\_stats(j + 1, "GPU", end\_clock(iter\_time), |
|  | prob\_mutation, prob\_crossover, pop\_size, max\_gen, ui\_seed, |
|  | ga\_seed, ui\_width, ui\_height, num\_elements); |
|  | } |
|  | gen\_log.write\_stats(-1, "GPU", end\_clock(total\_time), prob\_mutation, |
|  | prob\_crossover, pop\_size, max\_gen, ui\_seed, ga\_seed, |
|  | ui\_width, ui\_height, num\_elements); |
|  | gen\_log.end(); |
|  |  |
|  | cout << endl; |
|  | cout << "###############################################################################" << endl; |
|  | cout << "GPU - END" << endl; |
|  | cout << "###############################################################################" << endl << endl; |
|  | } |
|  |  |
|  | return 0; |
|  | } |