Московский Государственный Технический Университет им. Н. Э.

Техническое задание

на разработку системы моделирования

технических устройств «ПА10»

Научный руководитель

_____Маничев В. Б.

Разработано студентами группы

РК6-92:

Алиев Е. А.

Баженов С.В.

Шаломихин М.В.

Инв.№ подл	Подп. и дата	Взам.инв. №	Инв.№ дубл	Подп. и дата

1. Введение

Данный документ содержит исследование и первые предложения по реализации универсального программного комплекса для моделирования технических устройств смешанной природы «Программа Анализа», версии 10 (далее ПА10). Областью применения данного продукта является проектирование и анализ промышленных изделий и систем, объединяющих в себе функциональные элементы различного типа, начиная от систем защиты и контроля, электрических схем систем управления, до бытовой техники.

Данная версия комплекса изначально разрабатывается как ориентированная на визуальное проектирование, и должна отличаться дружественным интерфейсом, широкими функциональными возможностями, масштабируемостью и гибким механизмом специализации под конкретные задачи проектирования.

В предыдущих версиях комплекса разработчикам не удалось в полной мере реализовать вышеперечисленные требования к подобным системам. Версия 7 характеризовалась высокой функциональностью, возможностью расширения набора моделируемых объектов и гибкостью, за счет использования С-подобного языка моделирования. Однако, она была написана под DOS и к настоящему моменту уже морально устарела, т.к. ей недостает графического интерфейса пользователя и поддержки множества современных технологий, таких как клиент-сервер и COM.

Версия 9 была ориентирована на многоплатформенность и дружественность рабочей среды, но, в силу многих недостатков, не обладала достаточной гибкостью и функциональностью, а также, что самое главное, эффективностью проведения расчетов.

Таким образом, основной задачей при разработке версии 10 можно считать устранение изученных недостатков предыдущих систем, а также внедрение новых, более эффективных и точных средств математического моделирования.

2. Основания для разработки

Основанием для составления данного документа послужило задание на технологическую практику.

Основанием для разработки комплекса ПА10 можно считать отсутствие на рынке систем проектирования ни одного продукта, удовлетворяющего необходимым требованиям по надежности, стабильности и простоте использования. Как правило, нестабильной оказывается либо среда проектирования, либо используемая система проведения математических расчетов, либо и то и другое.

3. Назначение разработки

Система ПА10 предназначена для анализа динамики различных (механических, гидравлических, пневматических, тепловых и смешанных) технических систем, функционирование которых описывается системой ДАУ.

Одним из применений разрабатываемой системы является использование её для проведения лабораторных работ по курсу «Основы Автоматизированного Проектирования» с целью обучения студентов современным методикам проектирования технических устройств и анализа их динамических характеристик.

4. Требования к программе или программному изделию

4.1. Требования к функциональным характеристикам

При проектировании крупных технических систем, таких как реактивные двигатели, интегральные микросхемы, коммунальные системы зданий и проч. становится невозможным и нежелательным её рассмотрение на уровне отдельных функциональных элементов. Кроме этого, практически невозможно представить сложную систему в виде текстового описания, без применения визуальных средств разработки.

В таком случае основным объектом проектирования становится не функциональный элемент, а модуль.

В силу вышесказанного система ПА10 должна предоставлять проектировщику простой и доступный интерфейс, независимо от степени детализации. Моделируемый объект должен задаваться графическим изображением эквивалентной схемы, которая представляет собой совокупность связанных между собой по определенным правилам элементов, являющихся математическими моделями компонентов анализируемой технической системы.

По заданной эквивалентной схеме ПА10 должен формировать математическую модель в виде системы дифференциально-алгебраических уравнений (ДАУ), описывающей динамические процессы в исходной технической системе. После интегрирования ДАУ программный комплекс должен отображать полученные результаты в текстовой или графической форме.

4.2. Требования к надежности

Система ПА10 не должна анормально завершать свою работу при возникновении критической ошибки в одном из составляющих её модулей.

4.3. Требования к информационной и программной совместимости

Система ПА10 строится как кроссплатформенная. То есть должна функционировать на вычислительных машинах с ОС семейств Win32/64 и Linux.

4.4. Специальные требования

Система ПА10 должна быть построена таким образом, чтобы математический модуль был встраиваемым. Кроме того, у пользователей может возникнуть потребность к встраиванию в ПА10 собственных исполняемых модулей, переписать которые на внутренний язык ПА10 (или модифицировать их так, чтобы они могли воспринимать какой-либо жесткий интерфейс ввода/вывода в/из ПА10) он не имеет возможности.

Это выдвигает дополнительные требования к системе. Она должна реализовывать машинно- и операционно-независимый интерфейс подключения новых исполняемых модулей и изменения существующих. Так же в будущем может возникнуть потребность к встраиванию самой системы ПА10 в какие-либо программные комплексы. В качестве первого приближения на платформе Windows можно использовать технологию «Dll», а на платформе Linux – «Shared Object».

5. Состав системы

5.1. Общие соображения

Большие системы характеризуются сложностью связей между компонентами. Проблемным, как правило, является как изменение существующих, так и добавление новых модулей, т. к. необходима полная и подробная информация о системе в целом, в противном случае ее надежность резко падает.

Вследствие этого система ПА10 проектируется как объектно-ориентированная не только на уровне графического интерфейса пользователя, но и на уровне моделируемых элементов, методов вычисления и форматов хранения и передачи данных.

Соответственно, не должно существовать монолитного ядра, объединяющего в себе основную функциональность системы. Такая модель не только облегчает перенос ПА10 на другие платформы, но и позволяет вести параллельную разработку нескольких модулей. Основным требованием к ним будет являться прозрачное вхождение в уже работающую систему.

Помимо этого, объектная схема позволит в будущем физически разносить взаимодействующие модули на разные рабочие станции, как это предлагалось, но не было до конца реализовано в версии ПА8.

Само понятие модуля подразумевает не функциональное разделение системы на такие единицы, как «вычислительный» или «графический» модули, а составление ее из более мелких и более простых компонентов. Каждый компонент специализирует под свои нужды функции, предоставляемые ему другими компонентами и, в свою очередь, предоставляет им функции, требуемые от него. Если имеющиеся стандартные функции окружения не удовлетворяют компонент, он может их расширить или взять часть функций на себя.

Подобная архитектура уже используется в системах серии ПА для случая элементов электрических эквивалентных схем. Элемент сам вычисляет свою функцию приращения. Т.е. в системе не существует жесткого и заранее ограниченного набора элементов. Это позволяет легко расширять набор моделируемых объектов, предоставляя возможности по анализу практически любых устройств.

Имеющийся на данный момент модуль математических расчетов DMAN также дает возможность пользователю корректировать процесс вычисления (посредством вызова пользовательской функции), специализируя библиотеку под нестандартные задачи анализа.

Таким образом, компонентная архитектура позволит решить многие из поставленных задач.

5.2. Описание компонентной модели

Рассмотрим более детально предлагаемую структуру системы и механизмы взаимодействия ее частей. Структурное описание и диаграммы классов представлены в виде UML диаграмм.

5.2.1. Значение компонентов

Компонент в разрабатываемой системе играет ключевую роль. На него возложена обязанность не только по хранению параметров элемента схемы или целого модуля, вычислению переменных состояния, но и по реализации взаимодействия с пользователем, объектной базой данных, а также любым другим компонентом. На первый взгляд, это слишком нагружает его функциональность и усложняет проектирование. Но особенностью подобной парадигмы является то, что компонент не обязан удовлетворять всем выдвигаемым к нему требованиям сразу. Те действия, которые не были реализованы разработчиком, заменяются действиями по умолчанию. Это позволяет разработчику сконцентрироваться на решаемой задаче, а не на

удовлетворении всех причудливых требований системы. И в тоже время – модифицировать под свои нужды любые действия системы.

Простым примером компонентной модели может служить интерактивное телевидение. Обычное телевидение транслирует все, а для того чтобы воспользоваться его услугами нужно только включить телевизор. Но на него тяжело повлиять и любимый фильм легко пропустить. Это пример того, как взаимодействуют с компьютером старые устройства, и работает протокол передачи данных UDP.

Пункт проката видеокассет предлагает широкий выбор фильмов, кто угодно может придти и выбрать все, что ему хочется. Но для этого нужно купить карточку члена, оставлять залог и выходить из дома за каждым новым фильмом. Это пример клиент-серверной технологии.

Интерактивное телевидение позволяет проголосовать и выбрать тем самым желаемое развлечение. В голосовании можно не участвовать. Выбор ограничен, но зато он есть, и лишних действий делать не приходится.

5.2.2. Классы-обертки компонента

Согласно двум основным принципам объектно-ориентированного проектирования, хорошо построенная система должна обладать свойствами высокого зацепления и слабого связывания.

Под высоким зацеплением понимается специализация объекта проектирования (класса) в какой-то узкой области, реализация им какой-то конкретной деятельности, отсутствие распыления обязанностей, отказ от навязывания несвойственного ему набора функций.

Под слабым связыванием понимают как можно менее запутанное взаимодействие составляющих проектируемой системы. Чем меньше частей вступают в одновременное взаимодействие, тем проще понять систему, проще искать в ней ошибки, тем лучше, вообще говоря, продумана ее архитектура.

Поэтому компонент в системе ПА10 не является целостным объектом. Это означает, что с программной точки зрения он представляет собой совокупность объектов нескольких классов. Объединяет их то, что вместе они обслуживают один элемент эквивалентной схемы, блок визуализации результатов или целый модуль.

В центре этой группы стоит объект центрального класса компонента, реализующий непосредственную деятельность заключенного в компонент элемента, хранящий значения внутренних переменных состояния и методы, вычисляющие их конечные приращения. К примеру, для эквивалентной модели диода такой объект будет содержать значение проводимости элемента, а также метод, определяющий изменение тока, проходящего через элемент. В данном контексте нет никаких отличий между центральным объектом компонента в ПА10 и понятием элемента в предыдущих версиях системы ПА.

Предъявление к элементу требований по хранению, отображению и взаимодействию его с конечным пользователем, приводит нас к понятию классов-оберток. Класс-обертка компонента наделяет центральный объект определенной дополнительной функциональностью. Именно появление классов-оберток позволяет выделить центральный объект, вместе с обслуживающими его обертками, в отдельное понятие компонента.

Примером такого класса может служить графическая обертка на Рис. 1. Она наделяет компонент способностью отображаться в рабочем поле на экране компьютера. Под рабочим полем будем понимать отдельное окно, в котором содержится отдельный проект или составляющий его модуль. Подразумевается, что в программе одновременно может быть открыто несколько проектов.

Взаимодействие между центральным объектом компонента и его графической оберткой идет в одностороннем порядке. Это означает, что центральный объект не обладает никакой информацией об обслуживающей

его обертке и не вызывает ни один из ее методов, в то время как объект класса-обертки имеет почти полный доступ к параметрам работы и переменным (атрибутам) первого.

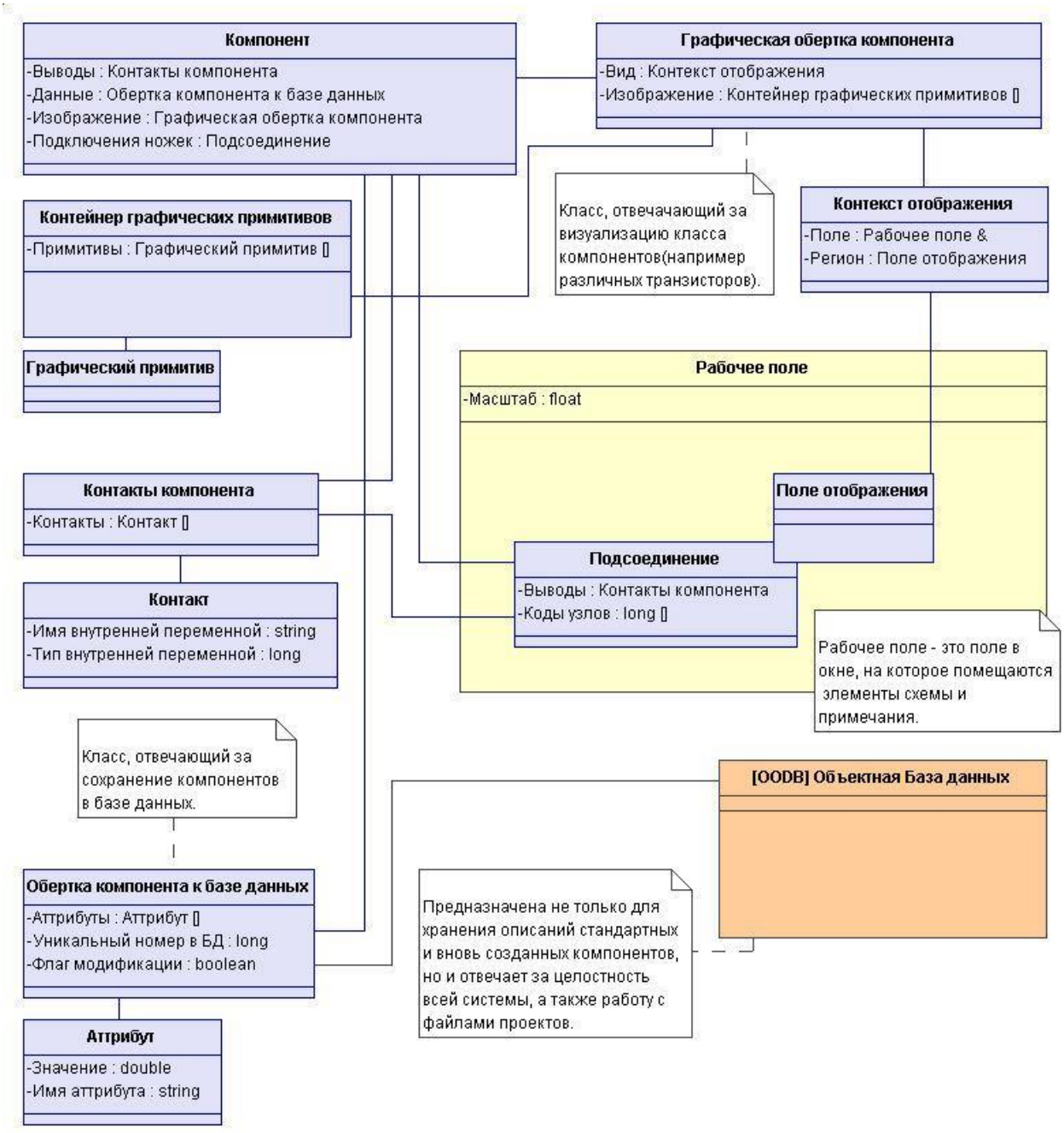


Рисунок 1. UML-диаграмма классов для оберток компонента

Графическая обертка написана специально для целей визуализации и «понимает» принятый интерфейс взаимодействия с графической подсистемой. Это облегчает проектирование и реализацию, поскольку не

смешивает такие различные по семантике действия, как расчет параметров и отображение условного обозначения. Кроме того, это позволяет подменять или совершенствовать по мере необходимости функции отображения, не меняя внутренней логики работы моделируемого объекта.

Отображение элементов можно реализовать и централизованно, однако такой подход имеет ряд недостатков. Помимо очевидной невозможности заранее предусмотреть внешний вид для всех будущих объектов, существует проблема навязывания проектировщику формата представления условных обозначений.

Статические изображения формата BMP, GIF или JPEG, очень легко включить, однако это неудовлетворительное решение. В ряде задач, таких как проектирование интегральных схем и гидравлика, требуется постоянный мониторинг состояния электрических выводов и положение вентилей, а также положение стрелок датчиков давления и температуры. Если заранее вкладывать в централизованную систему возможности по отображению подобных вещей, это не только затянет время разработки, но и чрезмерно усложнит программный код.

Графическая обертка компонента позволяет переложить на разработчика новых элементов и модулей обязанностей по реализации необходимой ему функциональности в плане графического интерфейса. Как обычно, обертку можно не реализовывать, в этом случае будет использована стандартная обертка с минимальными возможностями, отображающая, к примеру, просто прямоугольник.

Еще один положительный аспект применения классов-оберток заключается в возможности использования методик автоматического программирования, когда необходимый программный код создается под управлением, но без непосредственного участия человека.

Согласно Рис.1, условное обозначение в предлагаемой системе задается в виде изображения, построенного из отдельных графических примитивов. Это указывает пути к легкому проведению модификаций заданного изображения, таких как повороты, изменение масштаба, подсвечивание «красным» ошибочных или особо важных участков схемы, а также введение в изображение редактируемых полей, таких как наименование и основные характеристики, что избавляет от необходимости каждый раз входить в диалоговое окно.

Здесь важно отметить, что введение компонентов невозможно сделать «потом». Откладывание изменений в архитектуре на дальние этапы разработки приведет к тому, что уже написанный код окажется ненужным. Получить компонентную систему простой компиляцией из старых разработок и подходов заманчиво, но невозможно хотя бы уже потому, что добавление лишней кнопочки или строки состояния в систему со статической (не компонентной) организацией приводит к неизбежной перекомпиляции и пересмотру механизма вызовов.

5.2.3. Системо-независимые интерфейсы подсистем

Одним из заявленных требований к ПА10 было внедрение в нее возможности пополнять и развивать библиотеку элементов, создавать модели многополюсников, которые было бы легко использовать повторно уже в другом проекте. Из этого немедленно вытекает, что проектировщику необходимо разрешить самому задавать классы-обертки для отображения и хранения компонента, а также проведения каких либо дополнительных экзотических расчетов.

Таким образом, должен существовать устойчивый системо-независимый интерфейс для реализации основных функций системы, т.е. пользователю должны быть предоставлены функции для отображения и вывода в файл необходимых данных, одинаковые во всех инкарнациях

системы ПА10. Соответственно, должны существовать подсистемы, реализующие эти интерфейсы.

На данный момент можно выделить четыре таких подсистемы:

- Системо-независимая графическая подсистема;
- Системо-независимая подсистема сообщений;
- Системо-независимая подсистема взаимодействия с исполняемыми файлами;
- Системо-независимая подсистема работы с диском;

Графическая подсистема и подсистема сообщений могут быть реализованы либо с помощью библиотеки CLX Borland Software Corporation, либо посредством GTK+, распространяемой под лицензией GPL. Для применения CLX существует одно ограничение – ее нельзя использовать бесплатно в коммерческих целях. В тоже время, используя GTK+, необходимо предоставить в общедоступное использование исходные коды продукта.

6. Стадии и этапы разработки

6.1. Полнота системы

На раннем этапе нет смысла создавать программного монстра, который обладал бы полной функциональностью и воплощал в себе все, что можно ожидать от подобной системы проектирования. С другой стороны, разрабатывать легкий фантик, в котором можно было бы в скором времени покликать мышкой, нецелесообразно, поскольку он почти целиком пойдет в корзину, а научиться чему-либо на таком проекте тяжело.

Предлагаемая компонентная модель имеет то преимущество, что позволяет построить легко расширяемую систему. Это означает, что начав с малого, снижается вероятность того, что при переходе к большому возникнут непреодолимые трудности.

У системы, вскорости построенной при помощи средств быстрой разработки приложений, существует ярко выраженный потолок развития. Дело в том, что полностью полагаться на стандартные библиотеки, вроде MFC или VCL, при разработке системы, подобной ПА10, довольно наивно. Прежде всего потому, что они ориентированы на офисные приложения. При помощи таких средств можно довольно быстро накидать интерфейс пользователя, однако такая система будет вырвана из контекста и не сможет быть адаптирована под изменившиеся условия без пересмотра и перекомпиляции. Кроме того, они ничем не помогут в решении собственно проектных задач, возникающих при разработке САПР.

Есть еще одно важное замечание по поводу использования обычных распространенных средств. Без детально продуманной архитектуры и механизмов взаимодействия подсистем тяжело будет подключить сторонних разработчиков. Хотя, к примеру, Microsoft Visual C++ 6.0 и является хорошо известным средством, одного этого недостаточно. Если при разработке опираться только на него, то время, сэкономленное на ненужности изучения

среды, с лихвой окупится необходимостью полностью изучить внутреннее устройство ПА10, поскольку без этого нельзя будет ни добавить кнопочку, ни изменить графики, ни включить дополнительное поле в сохраняемые в файл данные.

Таким образом, несмотря на первоначальную легкость и простоту проектируемой системы, она должна изначально строиться как большая и расширяемая, хотя бы потому, что дополнительного времени и сил это не потребует, а в дальнейшем поможет сэкономить те же самые время и силы.

Следовательно, следует выбирать не распространенные, а наиболее подходящие средства и технологии.

6.2. Принципы представления данных

В проектной документации для предыдущих версий системы ПА рассматривалась возможность использования реляционной базы данных для хранения информации об элементах моделируемых объектов. Для версии ПА10 такой подход видится необходимым.

Использование в разрабатываемой системе компонентной модели с ориентацией на модульное представление проектируемых в ней устройств выдвигает новые требования к подсистеме хранения и представления внутренних данных. Поскольку система строится с учетом того, что модели проектируемых устройств будут содержать потенциально большое число элементов, использование базы данных призвано обеспечить прозрачное взаимодействие компонентов системы с необходимой им информацией.

Это позволит не только производить легкий поиск и модификацию используемых объектов, но также упростит построение и использование системы за счет абстрагирования самой информации от способа ее физического хранения и представления.

Пользователя не должно беспокоить количество и расположение файлов, составляющих проектируемую им модель. Кроме того, он должен

иметь возможность свободного переноса компонентов уже существующих устройств во вновь разрабатываемые.

Все эти возможности могут быть реализованы применением специальной базы данных.

6.2.1. Анализ возможностей

Сейчас наиболее часто используются реляционные базы данных (MS Sql Server, Oracle). Это обусловлено в первую очередь тем, что объектные базы данных заслужили репутацию медленных и ненадежных. Однако, это устаревший взгляд. На данный момент, производительность современных вычислительных систем позволяет использовать объектно-ориентированные СУБД в задачах, логика которых плохо интегрируется в реляционную модель.

При построении сложных иерархических систем часто приходится создавать объектно-ориентированный интерфейс к реляционной базе данных, что отнимает довольно много времени и сил, а также значительно снижает производительность, ради которой и используется реляционная база. Кроме того, подобным путем можно добиться лишь объектности, т.е. объединить данные в некоторые логические группы. В то время как объектная ориентированность подразумевает возможность задания гендерных отношений между схожими понятиями, поскольку простое отношение агрегации не охватывает всего множества возможных отношений между объектами.

6.2.2. База данных и компонентная модель

Компонентная модель, а также то, что под ней понимает Microsoft, реализуя технологию COM и ее расширения, использует динамическое связывание для вызова методов классов. Это значит, что функции связываются не статически по адресу, а по квалифицированному имени в момент запуска приложения. Данные о физическом расположении вызываемого компонента в случае с Windows находятся в реестре. Особенностью существующих

реализаций является предположение о том, что интерфейс всех функций компонента заранее известен. В этом случае становится тяжело работать при наличии нескольких версий компонента.

В .Net – компонентах данный недостаток устранен. Проблема была решена использованием метаданных, содержащих всю информацию об устройстве классов в компоненте. Выход тестовых версий .Net Framework под Linux ожидается к концу этого года. Однако из поддерживаемых средств разработки пока будет доступен только C#.

Проблема в том, что метаданные для CLR работают очень медленно. Для компонентов ПА10 наилучшим выходом будет создание собственной системы метаданных. На этом пути уже почти все реализовано и проведены первые испытания.

Поскольку без наличия компонентов и средств хранения данных об эквивалентной схеме невозможна дальнейшая разработка, к решению задач, связанных со специальной базой данных, следует приступить в первую очередь.

6.2.3. Объектно-ориентированная база данных

В ходе работы над проектом была разработана объектно-ориентированная база данных особого вида. Среди ее характерных черт можно выделить две: слабая, близкая к горизонтальной прямой, зависимость скорости добавления и поиска данных от количества записей, а также сравнительно низкий расход памяти. Приложение 1. содержит описание алгоритмов работы и графики испытаний данной разработки. Исходный код на C++ не включен в текущий документ, поскольку необходимости в этом не видно (документ и так велик).

Основными задачами в этом направлении можно считать повышение устойчивости работы базы на больших входных потоках данных, и разработка средств распределенного хранения.

6.3. Взаимодействие с конечным пользователем

С точки зрения компонентной модели, организация взаимодействия с пользователем также не должна быть централизованной.

На Рис. 2 представлена не слишком детализированная UML-диаграмма классов, призванных обеспечить взаимодействие с указателем мыши и клавиатурой. Самыми главными здесь являются обертка активного компонента и сервер сообщений. Графическая поддержка осуществляемых над компонентом манипуляций состоит в простой подмене некоторых графических примитивов, составляющих условное изображение компонента, или изменение их цвета.

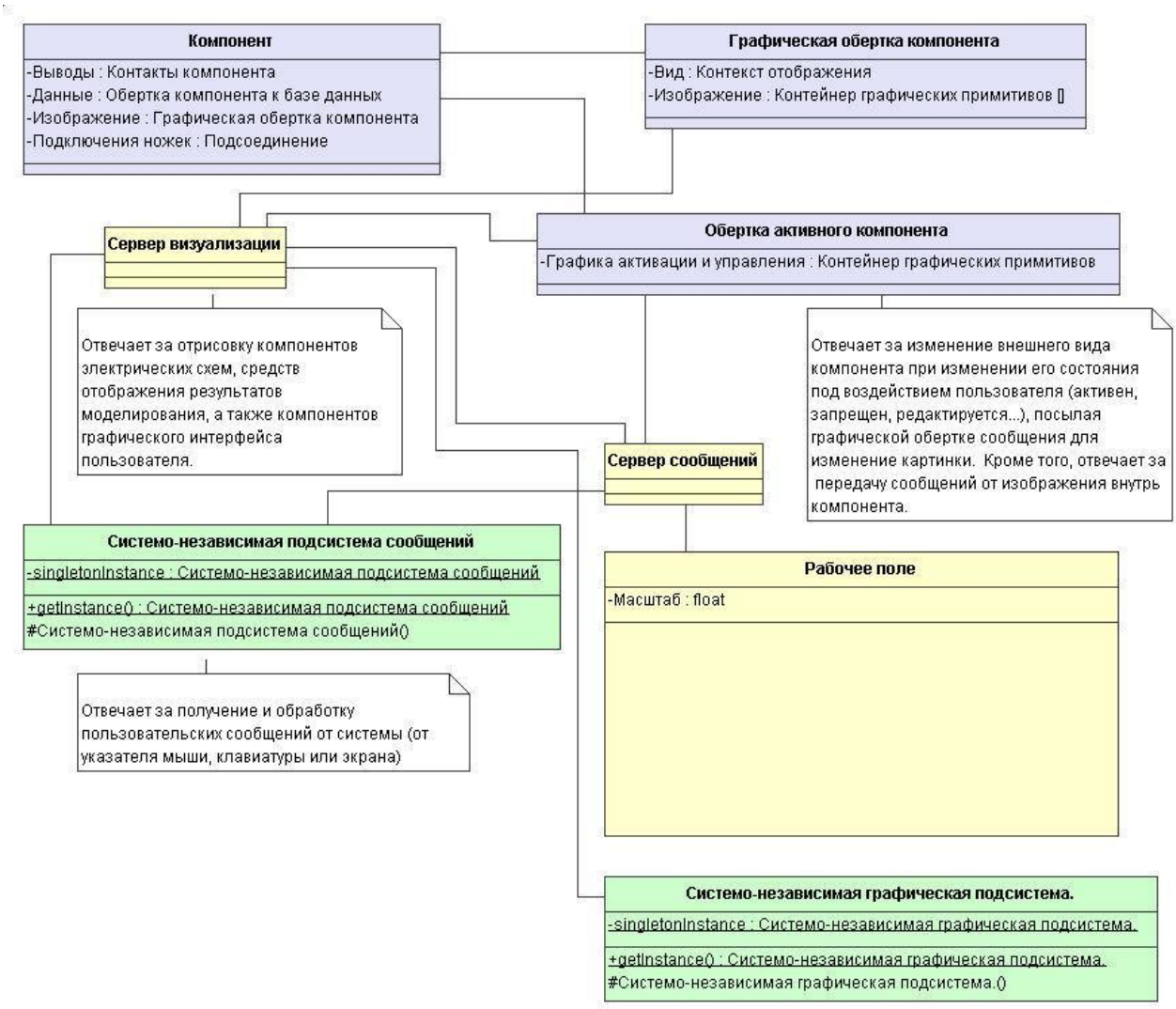


Рисунок 2. UML-диаграмма классов, обеспечивающих взаимодействие с пользователем.

Перерисовка осуществляется автономно и асинхронно сервером визуализации. Он узнает, какие части изображения испортились окнами других приложений, принимая сообщения от подсистемы сообщений, а какие части были обновлены самими компонентами, по сообщениям от контейнеров графических примитивов. Также он определяет порядок и время их перерисовки.

Сообщения о взаимодействии активных частей изображения передаются в компонент, поскольку именно разработчик компонента лучше всех знает, как на них реагировать.

Созданием системо-независимых подсистем и основных серверов следует заняться на втором этапе разработки.

6.4. Взаимодействие с исполняемыми файлами

Следующим этапом станет разработка механизма взаимодействия с исполняемыми файлами разных форматов на разных платформах. В данном направлении проведен анализ, результаты которого приведены в приложении 2.

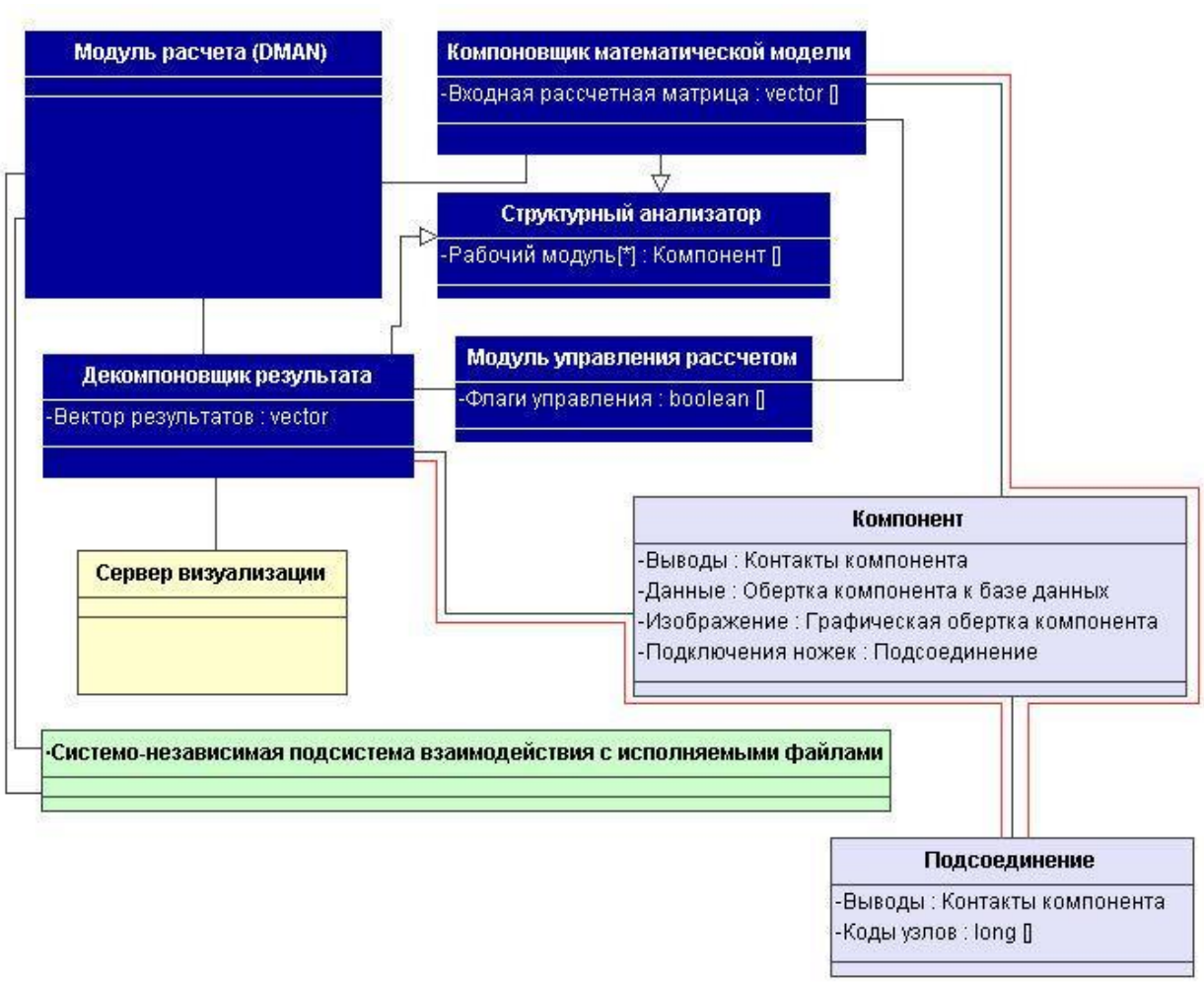


Рисунок 3. UML-диаграмма классов, позволяющих подключить модуль **DMAN**.

На Рис 3. изображена диаграмма взаимодействия классов, обеспечивающих данную функциональность ПА10. Первым испытанием для них станет обеспечение работы математического модуля DMAN.

Стрелки на диаграмме обозначают отношение наследование. Простые соединительные линии обозначают отношение простой ассоциации, и должны в дальнейшем выродиться в отношения агрегации или использования.

Компоненты, обозначенные как «компоновщик» и «декомпоновщик» предназначены для составления из данных о схеме матриц, подаваемых (получаемых) на расчет. Хотя само их составление будет производиться,

скорее всего, модифицированным узловым методом, наличие модульной абстракции не позволяет сделать это непосредственно.

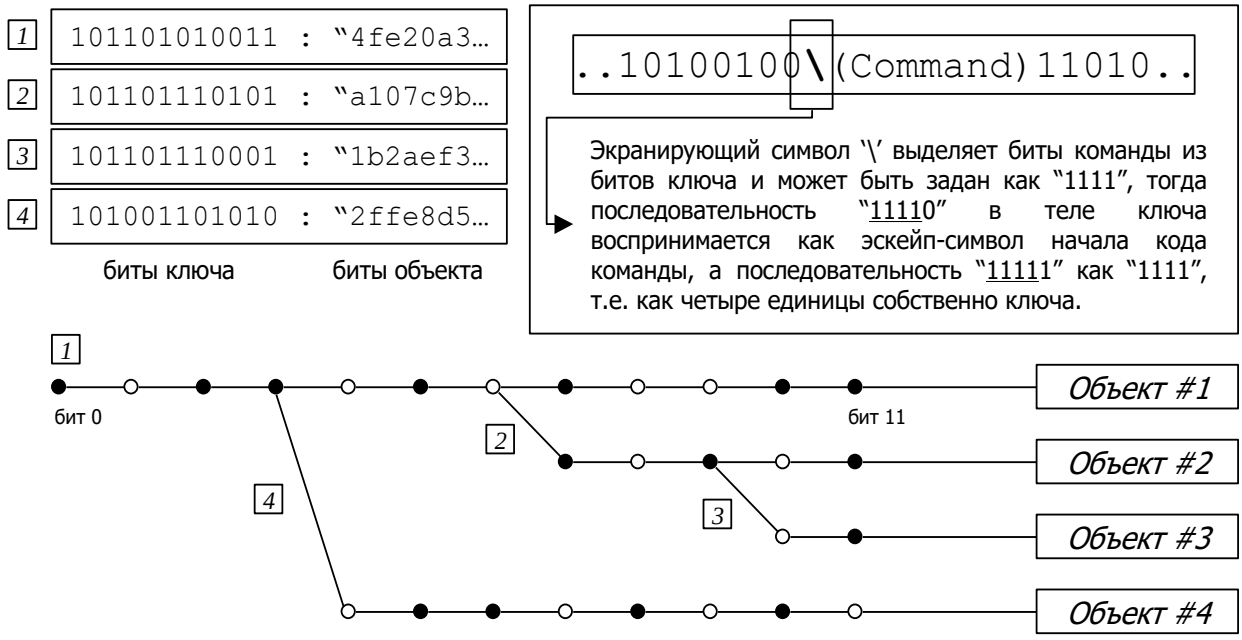
7. Приложение 1.

Реализация объектно-ориентированной базы данных.

7.1. Общая часть

В ходе работ над системой ПА-10 нами была разработана ООБД особого типа. Ее основу составляет бинарное дерево с ветвлением в расходящихся битах ключей. Алгоритм работы подобного дерева представлен на Рис. 4.

Схема хранения и поиска объектов по бинарным ключам



Оценка скорости поиска *S*: $\max S = O(N)$

$\text{avg } S$ – *зависит от набора ключей;*

Рисунок 4.

При добавлении новых ключей в подобное дерево, происходит их побитное сравнение с ключами уже добавленных объектов. В месте расхождения битов (т.е. в первом слева направо бите, в котором расходятся ключи) происходит ветвление и добавление управляющей команды, позволяющей сделать вставку и в дальнейшем ориентироваться по дереву.

При этом поддерживается не только быстрый поиск, но и составление произвольной иерархии.

Более полная информация по алгоритму работы данного модуля содержится в статье, представлявшейся на студенческой весне 2002 года, а также в документации к исходным текстам.

7.2. Проведенные испытания

На Рис. 5 приведены диаграммы добавления объектов по 32-х байтным ключам. Время добавления почти не зависит от размера базы, вследствие этого на графике представлен второй этап анализа – проверка скорости непрерывного внесения данных. Как видно из графиков, время одновременного добавления N объектов растет линейно от N, где N - число записей.

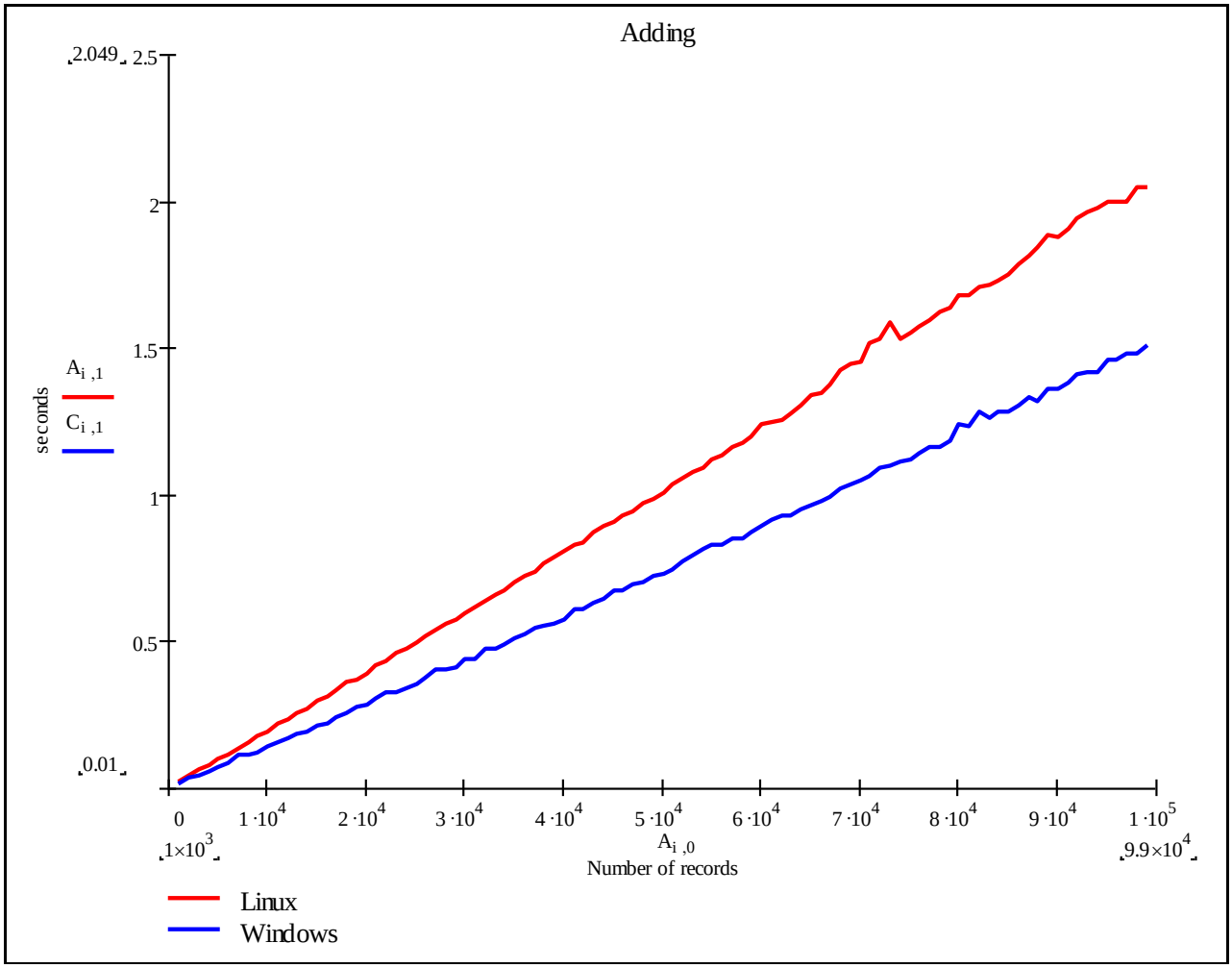


Рисунок 5. Добавление ключей в дерево.

Поиск ключа осуществляется проверкой существования всех ветвлений, присущих данному ключу. На самом деле, в оптимизированной версии побитной проверки не производится. Сравниваются блоки по восемь бит, существенно повышая скорость работы. Благодаря используемому алгоритму, скорость поиска записи в таком дереве зависит только от длины ключа и формально не зависит от общего числа хранящихся записей.

На Рис.6 приведены опытные графики зависимости скорости поиска от числа записей в дереве. На каждом шаге производился поиск 200 тысяч 32-х байтных ключей. Из графиков хорошо видно, что время поиска практически не зависит от объема базы.

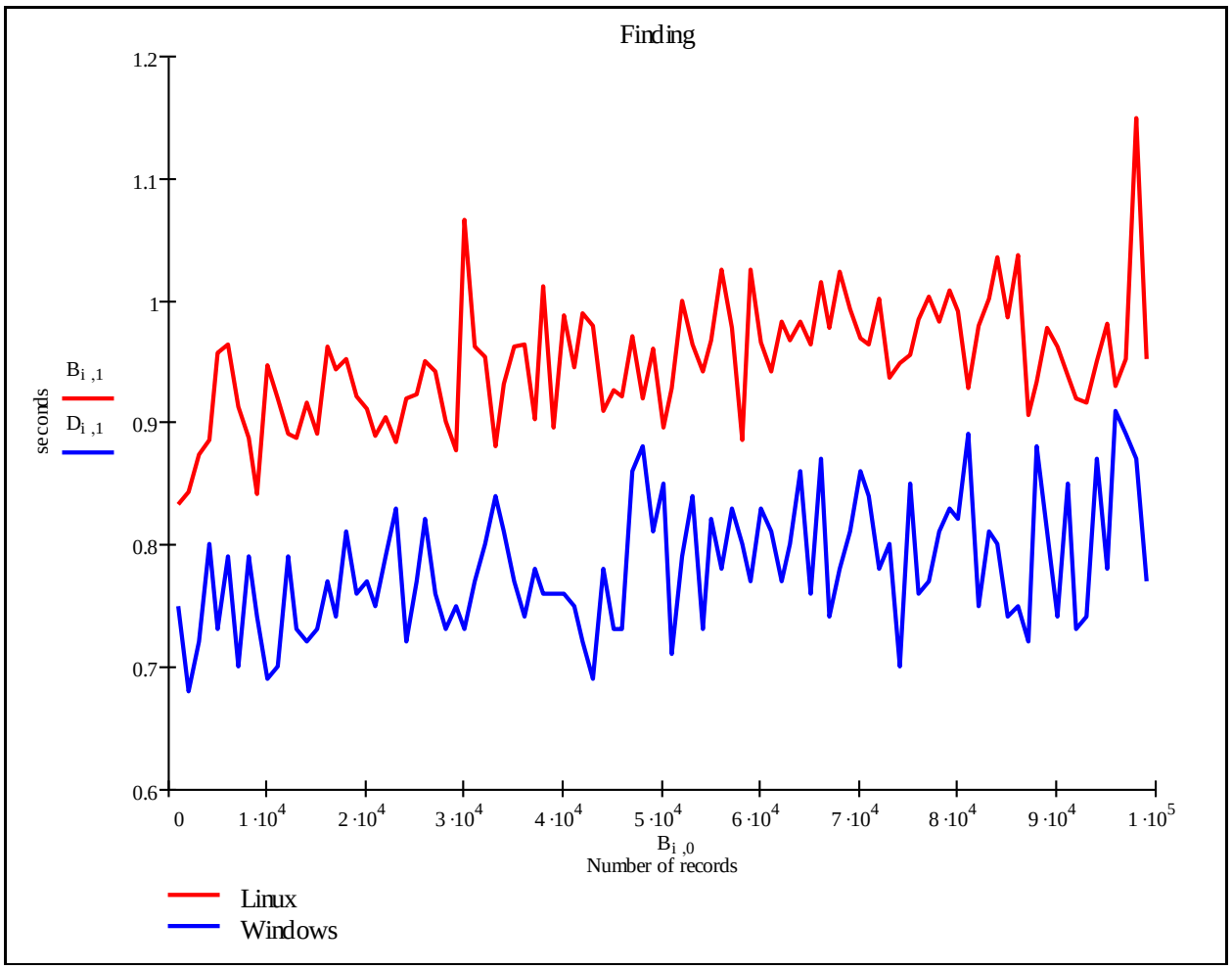


Рисунок 6. Поиск в дереве.

Их хаотическое поведение, скорее всего, объясняется тем, что современные ОС являются многозадачными и сложно предсказуемыми, т.е.

ОС может в любой момент времени начать работать со swap–файлом, что, естественно, сказывается на выполнении пользовательского приложения.

Используемая всей программой память достигала при этом 80-ти мегабайт. В первую очередь это связано с особенностями современных менеджеров памяти, которые не спешат освобождать возвращенное в систему свободное пространство.

7.3. Возникшие проблемы

Crush-тесты на Windows XP и Linux RedHat показали, что при постоянном добавлении/удалении тысяч элементов из текущей базы, системные менеджеры памяти начинают вести себя неадекватно. Чаще всего это проявляется в возврате ошибки при попытки изменить размер выделенного блока памяти. Возможно, это связано с используемыми алгоритмами выделения в куче, для нормальной работы которых кончаются свободные адреса, в то время как свободная память еще остается.

Так или иначе, в тесте, состоящем в добавлении и последующем освобождении N объектов по 32-х байтным ключам, а также повтору операции для N+1000, система выдавала критический сбой при распределении памяти, как только N поднималось от 1 000 до 198 000 записей. В том случае, если тест сразу начинался с 200 000 записей, никаких сбоев не возникало. Тестов по выявлению пиковой емкости хранилища не проводились.

Описанные выше особенности (быстрый поиск при таком же быстром добавлении) данных деревьев открывают широкие возможности при реализации эффективной и гибкой базы данных, тестовая версия которой уже реализована нами. Сейчас идет работа по повышению ее надежности и реализации всех функциональных возможностей, необходимых для внедрения данной разработки в систему ПА-10.

8. Приложение 2.

Технологии «Dynamic Link Library» и «Shared Objects».

8.1. Введение

Технологии «DLL» и «SO» применяются в ОС Windows и ОС Linux соответственно.

Данные технологии позволяют создавать динамические библиотеки объектов, подключающиеся к приложению на этапе выполнения, а не на этапе сборки.

Применение подобного подхода дает следующие преимущества:

- Возможность разработки приложений, на нескольких языках программирования одновременно. Функции, написанные на Fortran, могут с легкостью вызываться из программы на C++;
- Возможность модернизации программного продукта без перекомпиляции (заменой библиотеки).
- Экономия памяти. При запуске нескольких приложений, использующих общую динамическую библиотеку, в память загружается только одна ее копия.

8.2. Создание DLL и SO

Для создания dman.dll и dman.so использовались Visual Compaq Fortran (под Windows) и g77 - GNU Fortran Compiler (под Linux).

Шаги, требующиеся для создания библиотек:

Windows	**Linux**
• Создание Dll проекта. • Добавление ключей экспорта.	• Добавление ключей экспорта. • Копилирование библиотеки (ключ – shared) с помощью g77.

- Компилирование.

8.3. Особенности применения «DLL» и «SO»

8.3.1. Особенности применения Fortran функций в C++

Существует два важных аспекта применения Fortran функций в си-программах.

Во первых, следует обратить внимание на то, что в Fortran-е последовательное заполнение двумерного массива происходит не по строкам, а по столбцам.

Во вторых, в Fortran-е в функцию всегда передаются адреса переменных, а не их значения.

8.3.2. Особенности применения библиотеки DMAN в Windows и Linux

Различие в применении созданных библиотек заключается в первую очередь в различных методах доступа к динамическим объектам. Например загрузка Dll в Windows осуществляется функцией LoadLibrary(...), а аналогичная функция в Linux – dlopen(...).

Позиционирование ПА-10 как кроссплатформенной системы накладывает определенный отпечаток на ее разработку.

Уже на этапе проектирования требуется предусмотреть различия в использовании технологий «DLL» и «SO» и обеспечить легкий перенос программного кода, сохранив эквивалентность внутренней структуры системы для разных ОС.

С этой целью в структуру ПА-10 был введен платформо-независимый интерфейс доступа к вычислительному модулю, обеспечивающий общий интерфейс доступа к библиотеке DMAN, вне зависимости от операционной системы.