УДК 681.5:624.072.2-41

А.В. Аткин

ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ ДЛЯ РАСЧЕТА ПЛОСКИХ СТЕРЖНЕВЫХ СИСТЕМ НА ОСНОВЕ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПОДХОДА

Приведено описание основных процессов проектирования и реализации автоматизированной системы расчета сложных стержневых конструкций на прочность в линейной постановке. Основная функция системы — статический расчет плоских стержневых систем с большим числом элементов. Одновременно данная система должна служить тестовой платформой для проверки новых методов и алгоритмов, разрабатываемых в рамках научных проектов НОЦ «Строительная информатика». В связи с этим при ее проектировании уделялось особое внимание таким аспектам, как возможность добавления новых функциональных качеств; гибкость (возможность перестройки и внесения изменений в тело программы); возможность взаимодействия с другими программными средствами.

The author presents the description of the basic processes of design and implementation of an automated system for the linear strength analysis of complex plane grid systems. The main function of the automated system is to fulfill the static analysis of plane grid systems with a large number of members. Simultaneously, this system should work as a testing platform examining new methods and algorithms developed within the frames of the scientific projects at Scientific and Educational Center “Civil Engineering Informatics”. In this respect special attention was paid to such aspects as the ability to add new functional qualities, flexibility (ability to make reconfigurations and insert alternations into the body of the program), ability to interact with other software programs.

Проектирование автоматизированной системы (АС) является первым этапом ее разработки. Это итерационный процесс, где на каждой итерации могут выявляться новые требования, вноситься изменения и дополнения в проектируемую модель.

Проектирование АС включало следующие задачи:

- определение основных элементов системы и их взаимосвязей;
- проектирование структуры данных системы, включающей базу данных и модель стержневой конструкции;
- проектирование расчетного ядра, включающего «решатель» системы линейных алгебраических уравнений (СЛАУ);
- проектирование графической подсистемы;
- создание логической структуры системы;
- проектирование динамических аспектов системы;
- разработку структуры алгоритма.

Начальным шагом проектирования является формализация общих требований к системе. После анализа предметной области и задач, решаемых системой, были сформулированы основные требования к ней, включающие: требования к интерфейсу; требования к хранению данных; требования к ядру программы; требования к входным и выходным данным.

Далее, на основе сравнительного анализа возможных методов решения были выбраны: математический аппарат, лингвистическое и программное обеспечение. В качестве расчетного метода для написания ядра программы

выбран метод конечных элементов в форме метода перемещений. Для реализации системы выбран объектно-ориентированный подход, который позволяет реализовать описанные выше требования. В качестве языка проектирования был использован UML 2.0, а в качестве языка программирования — язык высокого уровня JAVA2, обеспечивающий устойчивость и гибкость разрабатываемой системы, а также ее независимость от платформы.

Определение основных элементов системы. На данном этапе были выделены элементарные части расчетной модели, которые в принятом объектно-ориентированном подходе представляют собой классы проектируемой системы. В качестве основных элементов системы были выделены: узлы (`Node`), стержни (`Bar`), закрепления (`Attachment`) и нагрузки (`NodeLoad` и `BarLoad`).

Базовым объектом является «узел» (`Node`). Атрибутами узла являются его координаты, наложенные связи, и приложенные в нем силы. Узел является независимым объектом и ничего «не знает» о стержнях, которые соединяют его с другими узлами.

Объект «стержень», в свою очередь, имеет ссылки на два узла, которые он связывает, хранит свои жесткостные характеристики, стержневые нагрузки и выполняет необходимые действия по обработке этих данных. На рис. 1 приведены графическая и проектная схемы стержня.

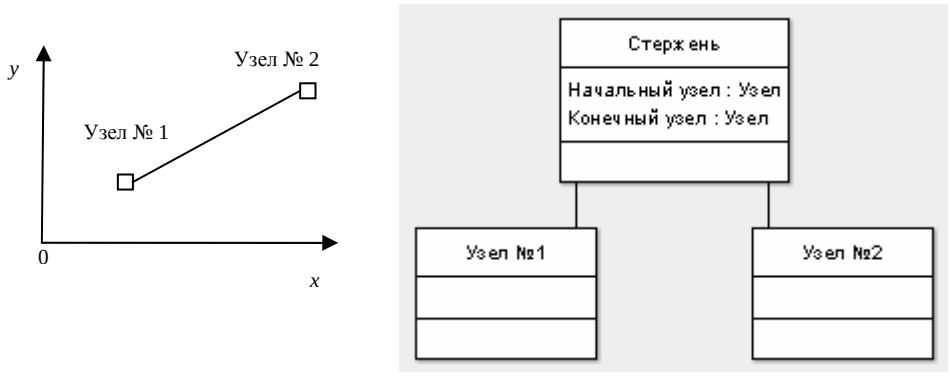


Рис. 1

Как видно из схемы, стержень различает начальный и конечный узел, что позволяет избежать ошибок при определении угла поворота стержня относительно глобальной системы координат. После создания стержня он должен автоматически определять свою длину и угол наклона.

Было принято решение использовать отдельные классы для таких элементов расчетной схемы, как закрепления и нагрузки. Использование классов вместо задания свойств объектов позволяет унифицировать процесс проектирования и программирования. Изменяя только внутреннюю спецификацию объекта, без изменения структуры передаваемых данных, можно обеспечить адаптацию и расширяемость данного класса. Например, при описании нагрузок, действующих внутри элемента (стержневых), использован принцип наследования. Запроектирован абстрактный класс стержневой нагрузки, являющийся родительским, от которого наследуются специальные классы на-

грузок. В родительском классе описаны общие методы обработки данных и отрисовки нагрузок. Все нагрузки, которые пользователь добавляет на стержень, хранятся в списке нагрузок, который определен классом-родителем. Схематическое представление этой структуры представлено на рис. 2.

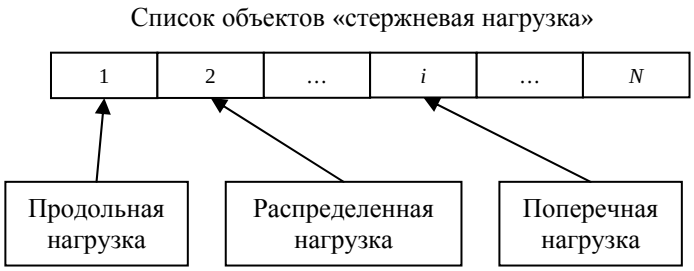


Рис. 2

Расширенная диаграмма взаимодействия описанных выше классов в нотации UML 2.0 приведена на рис. 3.

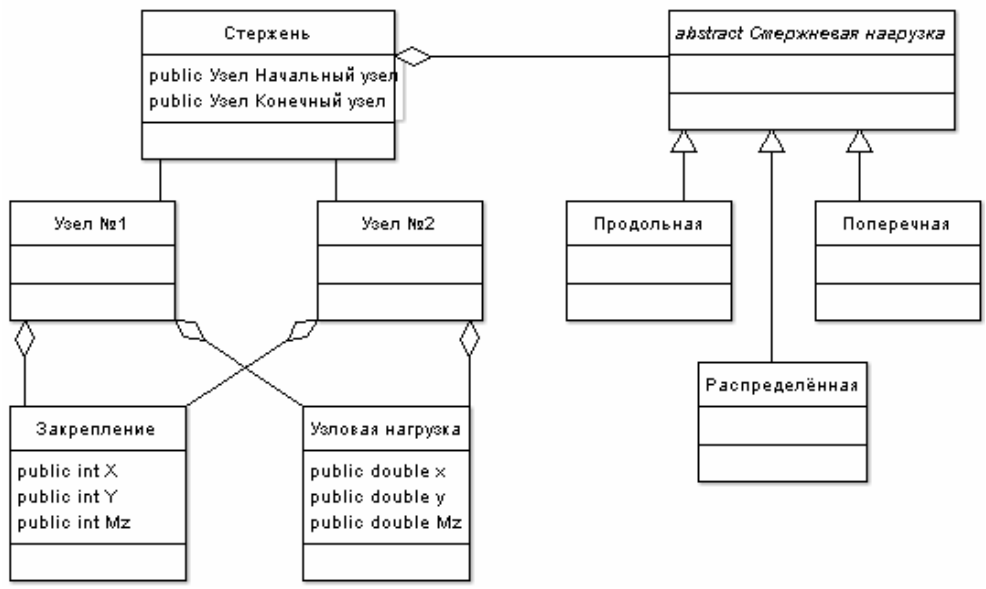


Рис. 3

Структура данных системы. Главным условием легко расширяемой системы является наличие стандартизованных структур данных, которые обеспечивают взаимодействие между разными частями программы. Для этой цели создан отдельный класс `Matrix`, на базе которого и происходит обмен данными между всеми частями программы. Этот класс инкапсулирует в себе двумерный массив как стандартную структуру данных, а также методы и операции работы с данными.

Модель стержневой конструкции. В качестве управляющего класса системы, описывающего модель конструкции, запроектирован класс «рама»

(`Frame`). Этот класс содержит в себе список узлов и стержней, матрицу жесткости, вектор перемещений и вектор нагрузок для всей системы. Он также реализует методы, с помощью которых внешние подсистемы взаимодействуют с обрабатываемыми данными. Класс `Frame` отвечает за взаимодействие разных подсистем: решения систем линейных алгебраических уравнений (СЛАУ), графического представления и интерфейса пользователя. В нем реализованы методы поиска узлов и стержней, выбора метода решения и предоставления результатов другим подсистемам. Все запросы на добавления, изменения или удаления элемента расчетной схемы происходят через него. Такая организация системы позволяет легко добавить или изменить метод расчета, для этого достаточно изменить внутреннюю структуру обработки данных, оставив интерфейс взаимодействия с внешними подсистемами без изменений. Схематическое представление этого подхода представлено на рис. 4.

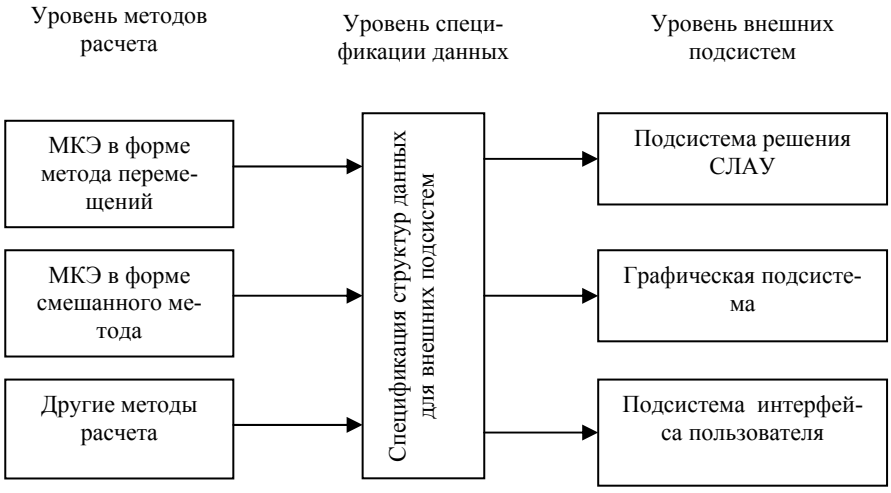


Рис. 4

Проектирование подсистемы решения систем линейных алгебраических уравнений представляет собой сложную научно-практическую задачу и было выделено в отдельный проект. Описанию этого проекта посвящена специальная статья.

Проектирование графических элементов и интерфейса пользователя. Механизмом управления, представляющим весь функциональный набор для пользователя, является главное окно. Этот графический объект представляет собой промежуточное звено между вводом данных, обработки вводимым/выводимых данных и графическим представлением расчетной схемы. Получая запросы от пользователя, эта подсистема направляет соответствующий запрос подсистеме, которая должна его выполнить. Например, если пользователь хочет удалить некий стержень, он отдает команду удаления, а главное окно переправляет этот запрос классу `Frame`, который и удаляет стержень.

Следующим шагом является проектирование графической подсистемы. Для этой цели создается графический холст, на котором отрисовывается расчетная схема. Этот холст отвечает также за функции масштабирования и

смещения схемы. Основной проблемой этой подсистемы является способ рисования элементов схемы, таких как: узлы, стержни, нагрузки, закрепления. В данной АС эта проблема решается при помощи принципа: «Каждый графический объект рисует себя сам». Для этого объекты должны реализовывать интерфейс рисования. Этого интерфейса нет в стандартной библиотеке, и он был создан разработчиками. Интерфейс включает только один метод, который называется `Draw` (рисование). Когда холст перерисовывается, он сообщает всем графическим объектам, что им надо нарисовать себя именно на этом холсте. Было принято решение, что эпюры моментов и деформированная схема будут отображаться в разных окнах, но начальный объект (холст) для всех окон один и тот же, что не создает никаких трудностей для рисования схемы в разных окнах. Диаграмма подсистемы отрисовки объектов приведена на рис. 5.

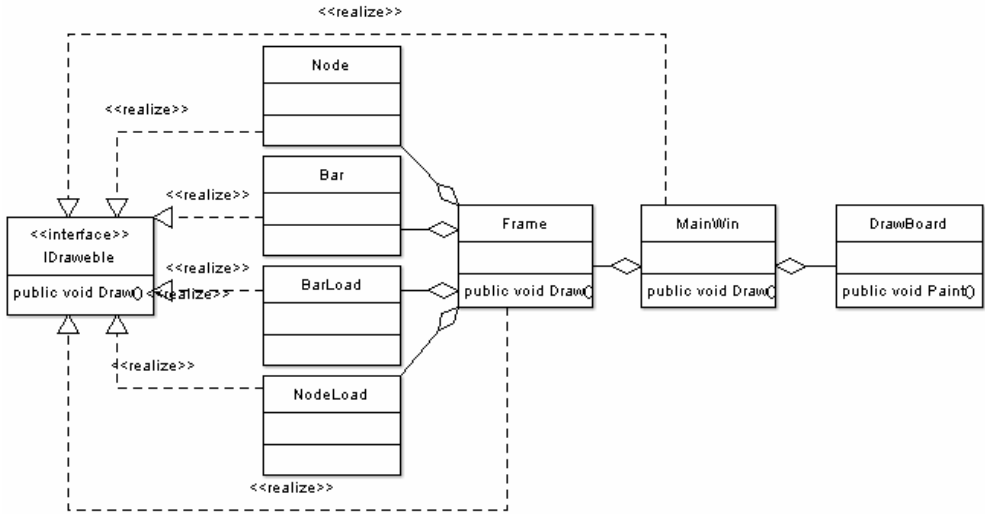


Рис. 5

Логическая взаимосвязь подсистем отображена на полной диаграмме классов (рис. 6). На этой диаграмме классы объединены в пакеты с целью чёткого отделения графических, расчетных и визуальных классов.

Проектирование динамических аспектов системы. В первую очередь строится диаграмма прецедентов или вариантов использования, которая облегчает понимание систем, подсистем или классов. Эта диаграмма предназначена для наглядного представления автоматизированной системы и выявления ключевых объектов, которые будут доступны пользователю. На этой диаграмме подсистемы специализированы по выполняемым ими функциям. Функциональные подсистемы программируются последовательно. Диаграмма прецедентов приведена на рис. 7.

Разработка структуры алгоритма. Структура алгоритма представлена в виде блок-схемы (диаграммы деятельности), на которой отображены все этапы решения задачи (рис. 8). Следует отметить, что в процессе реализации системы структура алгоритма претерпевала изменения и дополнения. Принятый в данном исследовании объектно-ориентированный подход позволяет легко адаптировать алгоритм к необходимым изменениям.

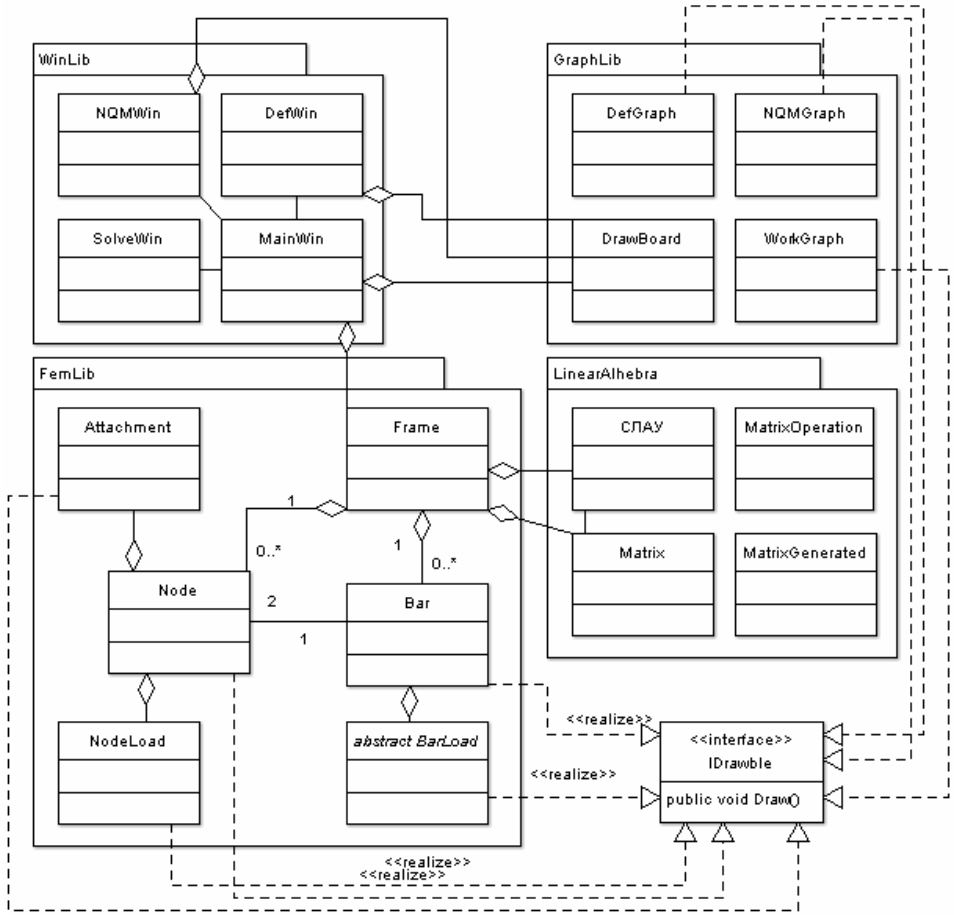

WinLib
NQMWin
DefWin
SolveWin
MainWin
GraphLib
DefGraph
NQMGraph
DrawBoard
WorkGraph
FemLib
Attachment
Frame
Node
Bar
NodeLoad
abstract BarLoad
LinearAlhebra
СЛАУ
MatrixOperation
Matrix
MatrixGenerated
1
0..*
1
0..*
2
1
<<realize>>
<<realize>>
<<realize>>
<<realize>>
<<interface>>
IDrawble
public void Draw()

Рис. 6

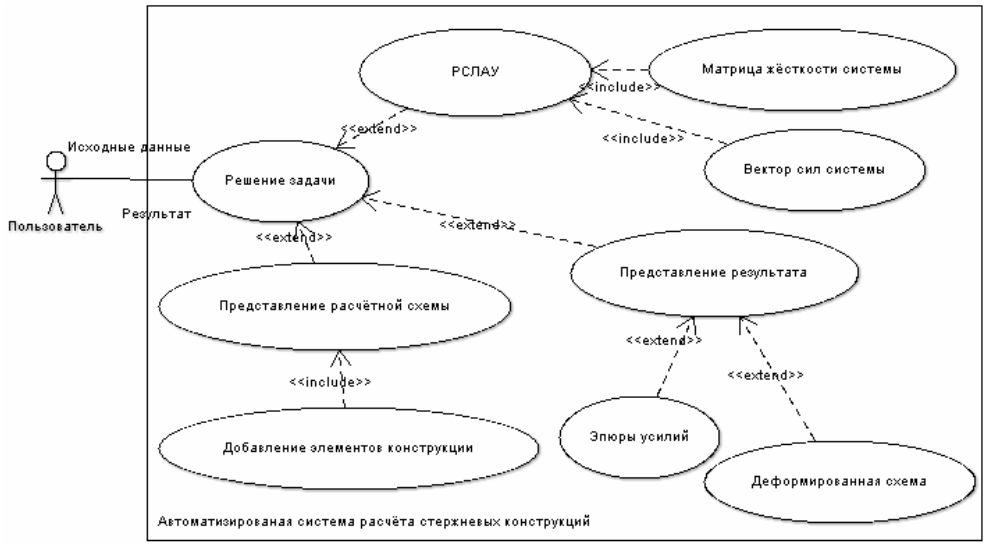

РСЛАУ
Матрица жёсткости системы
<<include>>
<<extend>>
<<include>>
Исходные данные
Решение задачи
Вектор сил системы
Пользователь
Результат
<<extend>>
<<extend>>
Представление результата
Представление расчётной схемы
<<extend>>
<<extend>>
<<include>>
Эпюры усилий
Добавление элементов конструкции
Деформированная схема
Автоматизированая система расчёта стержневых конструкций

Рис. 7

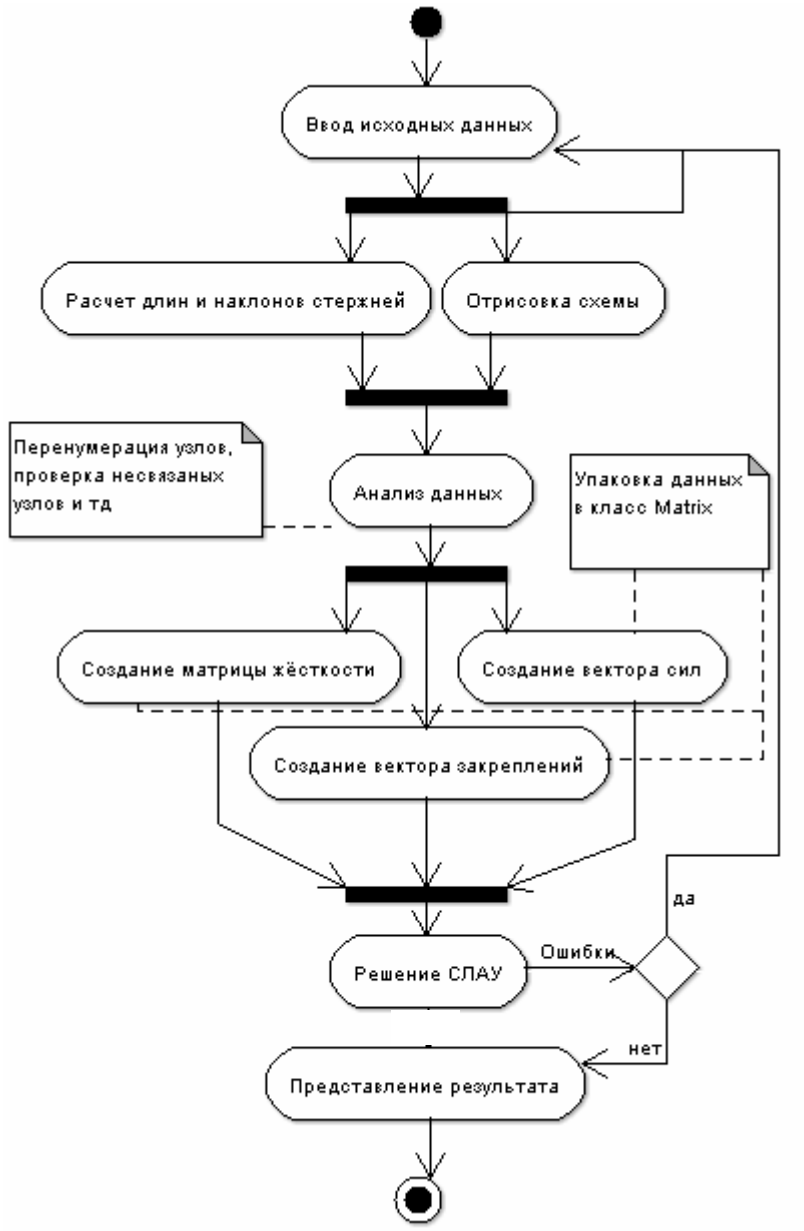


Рис. 8

Реализация автоматизированной системы. Как было отмечено ранее, для реализации системы был выбран объектно-ориентированный язык программирования JAVA 2.0. Выбор языка был продиктован следующими соображениями: его независимостью от платформы, свободным распространением, хорошим набором стандартных библиотек. Важным критерием явилось и то, что JAVA 2.0 используется международным сообществом ученых, работающих в области строительной информатики. При реализации системы по возможности использовались стандартные классы и структуры данных из множества библиотек, предоставляемых языком JAVA.

Программирование базовых классов системы. На рис. 9 приведены спецификации базовых классов `Node` (узел) и `Bar` (стержень).

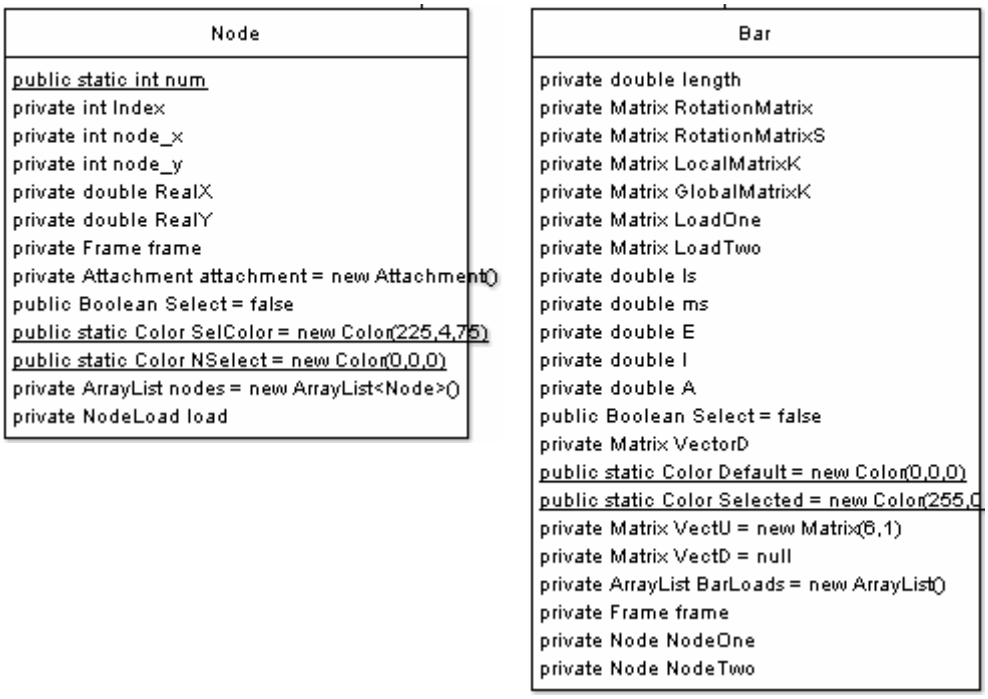


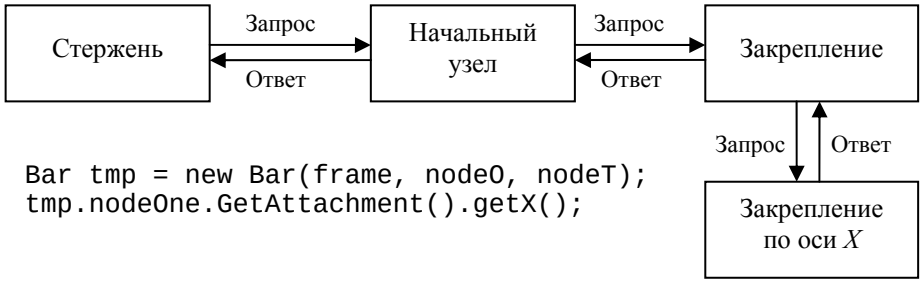
Рис. 9

Класс `Node` имеет такие атрибуты, как номер, координаты для рисования, расчеётные координаты, количество узлов, цвет, флаг выделения, классы закреплений и нагрузок, список узлов, с которыми этот узел связан, и ссылку на объект-родитель `Frame` (рама). Ссылка введена для того, чтобы не только рама могла взаимодействовать с узлами, но и узлы могли посылать раме сообщения. Также этот прием гарантирует принадлежность узла конкретному объекту `Frame`, что важно при возникновении исключительных ситуаций при загрузке или сохранении. Класс `Node` содержит список узлов, которые соединены с данным узлом, который необходим для реализации специальных методик, увеличивающих скорость расчета и сокращающих количество занимаемой памяти. Эти методики в настоящей работе не описываются.

Другим базовым классом является класс `Bar` (стержень). Он содержит в себе список нагрузок, ссылки на начальный и конечный узел, флаг выделения, жесткостные характеристики. В этом классе хранятся расчетные данные стержня: его матрица жесткости и векторы сил и перемещений. Этот класс также имеет ссылку на класс `Frame` (рама).

Классы `Node` и `Bar` достаточно просты по структуре, однако они являются основой автоматизированной системы. Для предотвращения сбоев в работе системы на нескольких этапах реализации этих классов предусмотрена возможность тестирования в режиме консоли. В качестве примера такого тестирования на рис. 10 приведена схема обращения к элементам узла через

стержень. Так как стержень содержит ссылки на узлы, то через них можно получить такие данные как закрепления или номер узла:



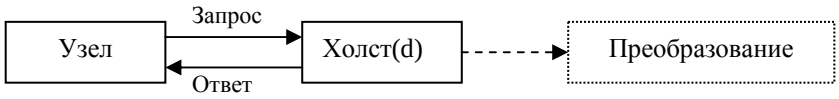

```
Bar tmp = new Bar(frame, nodeO, nodeT);
tmp.nodeOne.GetAttachment().getX();
```

Рис. 10

Программирование модели конструкции. Все узлы и стержни хранятся в двух списках, оба эти списка находятся в объекте рама. Список стержней имеет ссылки на объекты из списка узлов. На рис. 11 показано, как несколько объектов ссылаются на одни и те же данные в памяти.

Рис. 11

Особенности реализации интерфейса пользователя. Главным требованием к графическому интерфейсу пользователя является его интерактивность. Это требование реализуется через графический холст, основные принципы проектирования которого описаны ранее. В данной реализации схема конструкции может быть масштабирована и смещена относительно начальной точки. Пользователь имеет возможность выполнять определенные действия по щелчку мыши в зависимости от предустановленной команды. Для трансформации используются простые формулы. Когда какой-то графический элемент получает команду нарисовать себя на холсте, он получает преобразованные координаты и рисуется уже по ним. На рис. 12 можно увидеть, как происходит запрос и получение графических координат. Надо заметить, что реальные координаты не меняются и преобразование их не касается.




```
Node node = new Node(frame, 5,5);
node.Draw(d);
Point xy = d.GetWindowXY(node.x,node.y);
d.DrawRect(xy.x,xy.y,6,6);
```

Рис. 12

Программирование расчетной части системы. Как уже упоминалось, для этого был специально разработан класс `Matrix`. Был создан также вспомогательный класс со статическими методами для генерации матриц жесткости и трансформации для стержня по заданным жесткостным характеристикам и углам наклона.

В классе `Frame` происходит обработка списка стержней и выполняется генерация матриц жесткости и трансформации. Здесь же выполняется преобразование матриц жесткости стержней к глобальной системе координат. Выполнение стандартных операций над матрицами происходит во вспомогательном классе `MatrixOperations`.

Генерация матрицы жесткости системы выполняется следующим образом. Выделяется память под матрицу размерности [3*n* 3*n*], где *n* — количество узлов. Матрица жесткости стержня разбивается на 4 подматрицы-блока, и каждый блок, в зависимости от номера соответствующего узла, помещается на свое место в глобальной матрице жесткости. Перед этой операцией выполняется перенумерация узлов. На рис. 13 показан алгоритм создания матрицы жесткости для системы. Реализация этого метода на языке JAVA представлена на рис. 14.

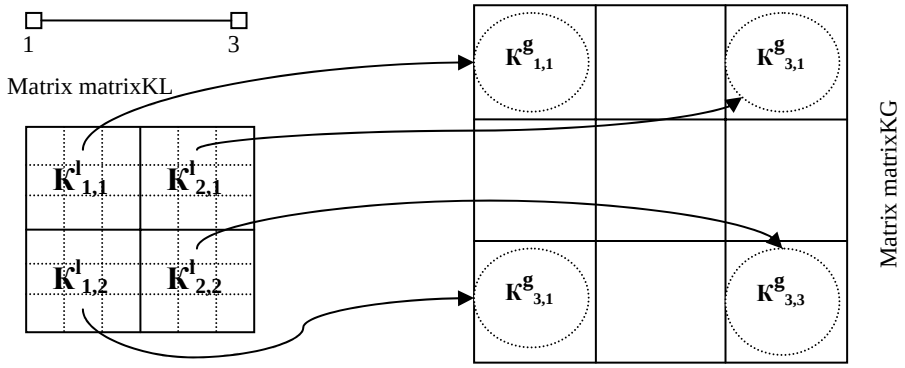


Рис. 13

По такому же методу индексирования создаются вектор сил и вектор закреплений. Последний вектор нужен для редуцирования матрицы жесткости и вектора сил. Принятый способ редуцирования включает следующий набор процедур: нахождение максимума в матрице жесткости; увеличение его в 2 раза; заполнение этим числом всех элементов, не входящих в результат; выделение результата.

```
matrixK = new Matrix(nodes.size() * 3, nodes.size() * 3);
int num1, num2;

for (int i = 0; i < num; i++) {
        num1 = bars.get(i).GetNodeOne().GetIndex();
        num2 = bars.get(i).GetNodeTwo().GetIndex();
        num1 = num1 * 3;
        num2 = num2 * 3;
        // ================================================

        //System.out.println(num1 + " ------------Стержень " + i
                //        + "--------------  " + num2);
        //bars.get(i).GetMatrixK().PrintMatrix(System.out);

        // -----------------Fist node---------------------
        matrixK.SetSubMatrix(num1, num1, bars.get(i).GetMatrixK()
                        .GetSubMatrix(0, 0, 3, 3));
        // -----------------Second node-------------------
        matrixK.SetSubMatrix(num2, num2, bars.get(i).GetMatrixK()
                        .GetSubMatrix(3, 3, 3, 3));
        // -----------------Second-first node-------------
        matrixK.SetSubMatrix(num2, num1, bars.get(i).GetMatrixK()
                        .GetSubMatrix(3, 0, 3, 3));
        // -----------------First-second------------------
        matrixK.SetSubMatrix(num1, num2, bars.get(i).GetMatrixK()
                        .GetSubMatrix(0, 3, 3, 3));

}
```

Рис. 14

После генерации данных для конечного расчета они передаются в подсистему РСЛАУ, которая возвращает результирующий вектор перемещений свободных узлов. Далее, с учетом вектора закреплений создается полный вектор перемещений, который и передается внешней системе.

Реализация функций загрузки и сохранения расчетной схемы. Данные функции реализованы при помощи встроенного механизма языка JAVA — сериализации объектов. Главным требованием этого механизма является то, что все зависимые объекты, так же как и основные объекты, должны реализовать интерфейс `Serializable`.

При реализации автоматизированной системы особое внимание было уделено созданию блоков кода, отвечающих за проверку правильности ввода/вывода и расчетных процедур.

Выводы. В данной работе описаны основные этапы проектирования и реализации автоматизированной системы, а также принципы и методы, использовавшиеся в работе. Описание завершается получением конечного результата в виде таблицы перемещений. Дальнейшее развитие системы, увеличение ее функциональности, развитие графического интерфейса, разработка визуального отображения результата рассматриваются в отдельных статьях, публикуемых совместно с данной работой.

В настоящее время коллективом авторов ведутся разработки в сфере оптимизации методов расчета и создания более продуктивных структур данных, которые позволят решать максимально сложные задачи с минимальными затратами вычислительных ресурсов.

Следующим этапом развития автоматизированной системы планируется реализация метода расчета для задач в нелинейной геометрии и создание графического интерфейса для пространственных задач.