В.А. Литвиненко, О.В. Рябов

ПРОГРАММНАЯ СРЕДА ВИЗУАЛЬНОЙ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ САПР*

Традиционный подход к созданию программного обеспечения (ПО) новой САПР предполагает его разработку на одном из языков программирования, с привлечением профессиональных программистов. Процесс программирования дорогостоящий, длителен и непредсказуем по сравнению с другими видами деятельности. Кроме того, разработанное программное обеспечение требует его тестирования, верификации, и апробации. Но все эти средства не гарантируют надежность программного обеспечения.

Другим подходом к разработке ПО САПР является эволюционный подход к созданию и развитию ПО САПР, основанный на модульном структуре ПО, постепенной адаптации или замене отдельных программных модулей проектных процедур и операций в зависимости от изменения технологии изготовления и элементной базы. При этом большое значение приобретает длительность жизненного цикла САПР и трудозатраты на адаптацию ПО САПР.

Альтернативным подходом является построение ПО систем без программирования. Для предметных областей с хорошо разработанными математическими методами и теорией вполне возможно отказаться от программирования. В качестве примера можно привести средства разработки СУБД Microsoft Access, работающие в графическом режиме, систему визуального программирования Borland C++ Builder и процессор баз данных (Date Base Engine), которые позволяют разработать базу данных и систему управлений к ней, не вводя программного кода. При этом разработчик не может допустить каких-либо ошибок в программном коде, например, опечаток, потому что весь необходимый код генерируется автоматически. Такая методика разработки возможна благодаря хорошей теоретической базе, на которой основаны реляционные базы данных.

В настоящее время теоретические основы САПР достаточно полно разработаны и формализованы [1], что позволяет начать разработку ПО САПР без написания программного кода. Можно выделить несколько способов такой разработки, которые, в общем случае, сводятся или к автоматической генерации кода из спецификаций, заданных пользователем, или объединение раннее созданных программных модулей по заданной разработчиком ПО САПР спецификации.

В статье рассматривается подход к построению ПО САПР на основе визуального проектирования. При таком проектировании ПО САПР системы достаточно выполнить несложные манипуляции мышью для размещения и конфигурирования элементов. Такая методология разработки ПО САПР позволит сократить цикл разработки и увеличить его предсказуемостью и формализованность, что снизит риски инвестиций в разработку САПР.

Такая система должна состоять из модулей, выполненных в виде отдельных файлов содержащий исполняемый код. Требование размещения модулей в отдельных файлах обусловлено необходимостью легкого добавления, замены или удаления отдельных модулей. Программные модули могут образовывать многоуровневую структуру (рис.1).

Модули уровня n являются более полными и соответственно более сложными чем уровня n+1. При этом модули уровня n образуются из модулей уровня n+1.

* Работа выполнена при поддержке РФФИ (грант №05-08-18115)

Допустимо формировать модули из более простых уровней, чем следующий по порядку, т.е. для формирования уровня n можно использовать модули уровня равного или большего n+1, но следует избегать использования модулей уровня равного или меньшего n. Это связанно с необходимостью избежать циклической зависимости между модулями.

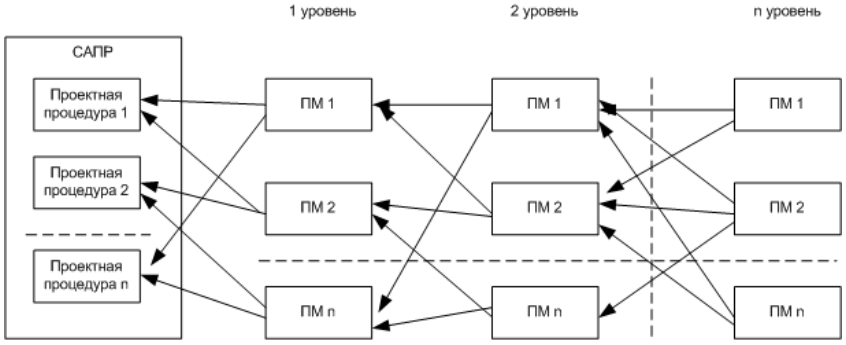

Рис.1. Структура модульной САПР

Модули можно реализовать в виде отдельных исполняемых файлов, передавая в качестве аргумента имя файла данных, или в виде динамически подключаемых библиотек, экспортирующих необходимые функции. Работу с динамически подключаемыми библиотеками, поддерживают практически все современные операционные системы, включая семейство MS Windows.

Первый вариант реализации исполняемых модулей позволяет эффективно организовать поддержку модулей лишь первого уровня. Поэтому для многоуровневого построения ПО САПР использование динамически подключаемых библиотек более предпочтительно.

Для таких САПР необходимо использовать унифицированные формы хранения и обработки данных в рамках системы модулей, а также предусмотреть проверку соответствия прототипов функций и соответствия типов операций выполняемых этими функциями. Для систем с большим числом модулей типы операций имеет смысл сделать иерархическими.

В целом требования к такой системе и программным модулям можно определить следующим образом:

1) модули должны использовать унифицированный, в рамках конкретной системы, формат данных;
2) система должна проверять соответствие прототипов функций и типов операций ими выполняемых;
3) система должна позволять визуальное управление модулями, что подразумевает различные операции определения порядка выполнения модулей, взаимосвязи между модулями, и т.д.;
4) не должно быть циклических зависимостей между модулями;
5) система должна показывать разработчику ПО САПР информацию о каждом модуле, такую как название модуля, подробное описание модуля, информация о требуемых модулях более низкого уровня, и, возможно, еще какую либо дополнительную информацию;
6) модули должны предоставлять информацию о себе, название, описание и пр.

Нецелесообразно создавать число уровней модулей больше трех или четырех. Это связанно с тем, что отдельный модуль такого уровня по функциональности

будет приближаться к функциональности отдельного оператора языка программирования высокого уровня (ЯПВУ). В связи с чем, потребуется организовать управление этими модулями аналогичное управлению операторами ЯПВУ.

С целью демонстрации визуального проектирования ПО САПР была разработана учебно-исследовательская (УИ) САПР, позволяющая формировать маршрут проектирования из модулей путем визуального размещения их в окне программы. Модули, в свою очередь, могут быть скомпонованы из подмодулей. Структурная схема УИ САПР показана на рис.2. Было реализовано два уровня модулей.

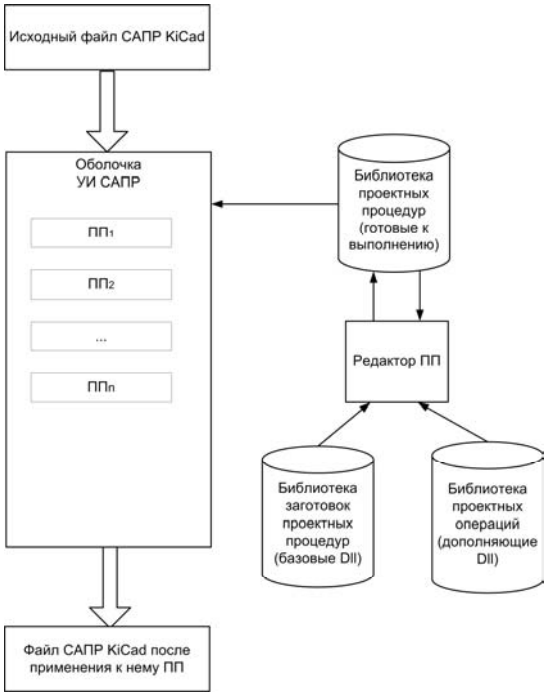

Рис.2. Структура УИ САПР

В качестве входных и выходных форматов файлов для учебно-исследовательской САПР был выбран формат, совместимый с одной из промышленных САПР KiCad [2], выбор которой был определен распространением ее под лицензией GPL [3], что позволяет свободно использовать САПР KiCad, получать доступ к исходным текстам программы, и при необходимости вносить изменения в исходные тексты с правом дальнейшего распространения.

KiCad - это кросс-платформенный, распространяемый по лицензии GPL программный комплекс класса EDA с открытыми исходными текстами, предназначенный для разработки электрических схем и печатных плат. Поддерживаются операционные системы Linux, Windows, FreeBSD и Solaris. Разработчик САПР KiCad - Жан-Пьер Шарра (фр. *Jean-Pierre Charras*), исследователь в LIS (фр. *Laboratoire des Images et des Signaux* — Лаборатория Изображений и Сигналов) и преподаватель электроники и обработки изображений в *IUT de Saint Martin d'Hères* (Франция).

Для работы с файлами САПР KiCad на языке C++ был разработан класс pcb. Класс содержит в себе всю информацию о плате, метод read(), производящий син-

таксический разбор файла, метод get(), кодирующий информацию о плате в формате KiCad с целью записи обработанной схемы в файл.

Для создания УИ САПР была разработана оболочка с возможностью визуального размещения модулей в окне и их последующего выполнения. Главная форма этой оболочки представлена на рис.3.

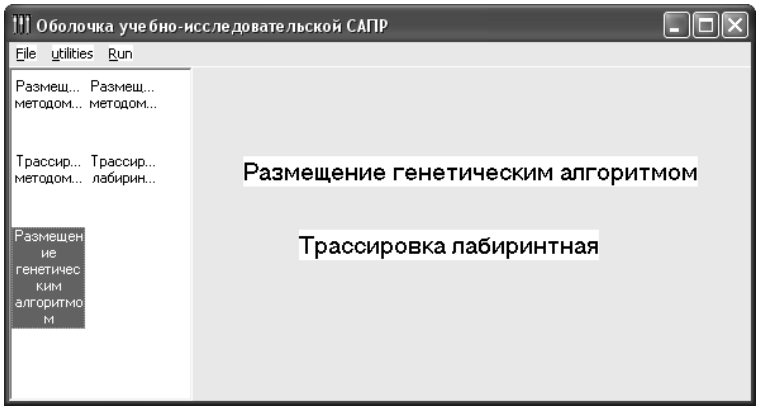

Рис.3. Окно оболочки

На окне оболочки размещены графические элементы, содержащие название программных модулей. Каждый такой элемент соответствует одному модулю. В правой части формы располагаются элементы, соответствующие программным модулям, реализующим алгоритмы, которые будут выполнены при выборе пункта меню «Run». В левой части формы размещены доступные модули, список которых генерируется автоматически на основание обнаруженных программой модулей. Пользователь может свободно перетаскивать их мышью по экрану программы, устанавливая в нужную позицию.

В состав УИ САПР входит программа, предназначенная для создания файлов модулей проектных процедур, в которой используются базовая dll и дополняющия dll.

Базовая dll - программный модуль, в котором находится программная реализация одного из алгоритмов САПР. Функции алгоритма, допускающие инвариантность, не включаются в базовую dll, а импортируются из дополняющих dll. Например, размещение элементов, размещаемых на печатной плате методом генетического поиска.

Дополняющая dll - это программный модуль, в котором находится программная реализация одной из функций, предназначенных для использования базовой dll. Например, оценка оптимальности размещения элементов на печатной плате.

Таким образом, разработанная среда визуальной разработки ПО САПР на основе промышленной САПР KiCad, позволяет визуально производить формирование маршрута проектирования из модулей, ассоциированных с алгоритмами проектной процедуры и алгоритмами проектных операций САПР.

Достоинством рассмотренного подхода к разработке ПО САПР является то, что для реализации какого-либо алгоритма не требуется разрабатывать код чтения и записи какого-либо файлового формата данных, эта функция предоставляется оболочкой. Также не требуется разрабатывать алгоритмы других проектных процедур, входящих в маршрут проектирования, для проверки функционирования нового алгоритма, поскольку их можно «набрать» из существующих модулей других

проектных процедур или использовать их программные модули непосредственно самой промышленной САПР (в данной УИ САПР -KiCad).

Кроме того, можно изучать альтернативные реализации алгоритма проектной процедуры, не разрабатывая новую программу, а лишь реализуя отельные функции. Для демонстрации такого подхода разработаны несколько альтернативных программных модулей реализации алгоритмов получения псевдослучайных чисел, подставляя которые визуально в генетический алгоритм размещения, можно провести анализ влияния различных алгоритмов генерации псевдослучайных чисел на проектируемую печатную плату. Состав альтернативных программных модулей может быть дополнен новыми модулями.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Норенков И.П. Разработка систем автоматизированного проектирования. – М.: Изд-во МГТУ им. Н. Э. Баумана, 1994. – 207 с.
2. http://www.lis.inpg.fr/realise_au_lis/kicad/index.html
3. http://www.gnu.org/licenses/gpl.html
4. http://ru.wikipedia.org/wiki/Kicad

Г.М. Касаткин, А.Ю. Таможникова

ПРИБЛИЖЁННОЕ РЕШЕНИЕ ЗАДАЧИ О КОММИВОЯЖЁРА

Введение. Для обнаружения неисправностей электрооборудования, непрерывно находящегося в эксплуатации, проводят контрольные осмотры и периодические технические обслуживания. Поскольку контрольных точек, как правило, несколько (n >2), то суммарные затраты на их контроль и обслуживание различны. Таким образом, приходим к необходимости решения оптимизационной задачи по минимизации суммарных затрат.

В качестве затрат могут выступать либо суммарные материальные затраты на средства технического обслуживания, либо суммарное время простоя системы. Решение задачи формально сводится к, так называемой, задаче о коммивояжёре.

Суть решения задачи о коммивояжёре состоит в сокращении перебора возможных последовательностей обхода ограниченного множества из N городов. Эта одна из наиболее известных задач исследования операций была сформулирована К. Менгером следующим образом [2]: найти наикратчайший маршрут, проходящий по одному разу через каждый из заданных городов; взаимные расстояния между городами известны.

Формально задача матрица $\left\|C_{ij}\right\|, \quad i \neq j \ (i, j = 0,1,2,3,...,n)$, где C_{ij} - затраты на перемещение между городами i и j. В роли затрат C_{ij} могут выступать: пройденное расстояние, время нахождения в пути от города i до города j, стоимость проезда от i до j и т.д. В общем случае может быть, что $C_{ij} \neq C_{ji}$.

Требуется найти перестановку $\Pi(i_1, i_2, i_3,..., i_n)$ чисел 1,2,...,n, минимизирующую суммарные затраты $F_n(i_1, i_2,..., i_n) = C_{01} + C_{02} + ... + C_{in}$.

При этом функционал $F_n(i_1, i_2,..., i_n)$ задается не только на перестановках всех n объектов, но и на перестановках, состоящих из любого числа элементов.