6. Программное обеспечение решающей системы “принятия решений”

6.1 Формирование пакета прикладных программ ППП “принятие решений”

Согласно определению САПР как инструментария проектировщика [1], ППП “принятие решений” является инструментом пользователей, которые не являются ни профессиональными программистами, разработчиками ППП, ни глубокими учеными-исследователями теории системного анализа, управления, численных методов и т.п. Поэтому формирование рассматриваемого ППП для “Решателя” (см. 1 рис. 10) состоит в разработке способов реализации математических моделей, методов и критериев в зависимости от поставленной задачи и интерфейсов с пользователями,- ЛПР. Возникающие при принятии решений задачи разделяются на “шаблонные” и новые еще не известные. В первом случае существует последовательность процедур их решения и соответствующие программы, реализующие изложенные в разделах 1-5 модели, методы и алгоритмы. Во-втором случае необходимо разрабатывать новые программы (приложения к известным средствам программирования) и соответствующие интерфейсы с ЛПР.

Обратимся к основной схеме, приведенной на рис. 10 главы 1и рассмотрим эти два подхода для реализации “Решателя”. При решении шаблонной задачи разработчик ППП может воспользоваться уже имеющимися программными продуктами на рынке, при этом он принимает на себя ответственность в правильности программной реализации и как бы принимает на “веру” методы, запрограммированные в ядре этих программных продуктов. Возможности протестировать ядро таких продуктов или изменить его для решения задачи отсутствуют, из-за конкуренции программных систем между собой. К таким программам можно отнести широко известные пакеты MatLab, MathCAd, Labview, Maple и другие. Но если приведенные программные системы не могут решить даже шаблонную задачу, то разработчику требуется создать свою программу и выполнить ее реализацию в “Решателе” на языках высокого уровня. При этом разработчик должен доказать правильность программной реализации, внести изменения в ядро разработанной программы. Это открывает дополнительные перспективы для разработки отдельных специализированных программ. Результаты работы “Решателя” должны быть сохранены в единой базе данных, соответствующие интерфейсы также должны быть разработаны или присутствовать в программном продукте. Рассмотрим процесс формирования ППП “Принятие решений”.

Процесс разработки ППП охватывает три основные области:

1. подготовка исходных данных, алгоритмизация изложенных в разделах 1-5 моделей, методов, функционалов;
2. проектирование и разработка программных модулей, экранных форм, отчетов, которые будут обеспечивать выполнение запросов к данным;
3. учет конкретной среды или технологии, а именно: топологии сети, конфигурации аппаратных средств, используемой архитектуры (файл-сервер или клиент-сервер), параллельной обработки, распределенной обработки данных и т.п.

Можно выделить следующие основные подходы к процессу разработки ППП.

Каскадная разработка или модель водопада (англ. waterfall model[6]) — модель процесса разработки программного обеспечения, в которой процесс разработки выглядит как поток, последовательно проходящий фазы разработки программы.

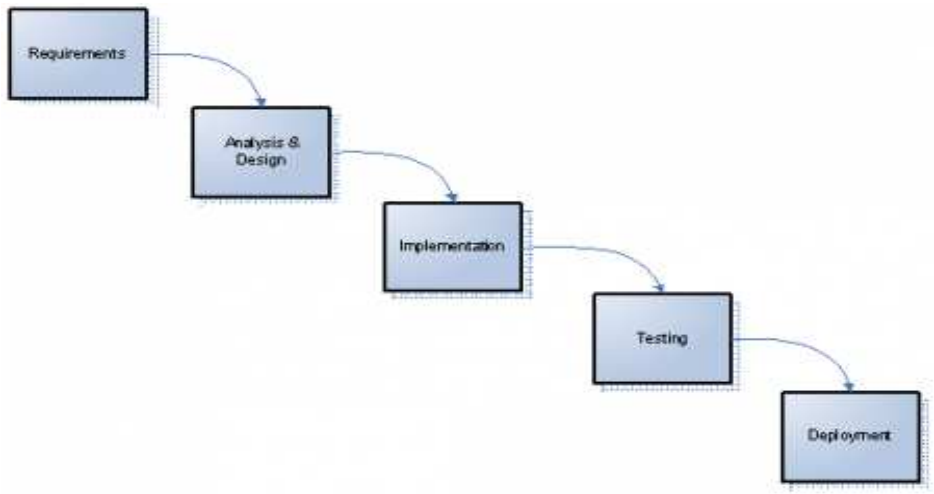

Рис. 47. Модель водопада разработки программного обеспечения

На рис 47. requirements – требования, analysis and design – анализ и дизайн, implementation – реализация, testing – тестирование, deployment – развертывание, внедрение

Эта модель предполагает последовательное исполнение следующий этапов:

1. Разработка требований (requirements): сбор требований заказчика программы и их преобразование в функциональные требования к программному продукту.
2. Анализ и дизайн (analysis and design): разработка модели предметной области (domain model), проектирование схемы базы данных, объектной модели, пользовательского интерфейса и т.п.
3. Реализация (implementation): создание продукта по спецификациям, разработанным на предыдущем этапе.
4. Тестирование (testing): включает проверку соответствия функциональности программного продукта потребностям пользователей (validation), а также поиск дефектов в реализации.
5. Развертывание (deployment): обучение пользователей, инсталляция системы, перевод в промышленную эксплуатацию.

Однако такая модель показывает себя не лучшим образом в проектах по разработке программного обеспечения с нечеткими требованиями, или требованиями, меняющимися по ходу разработки.

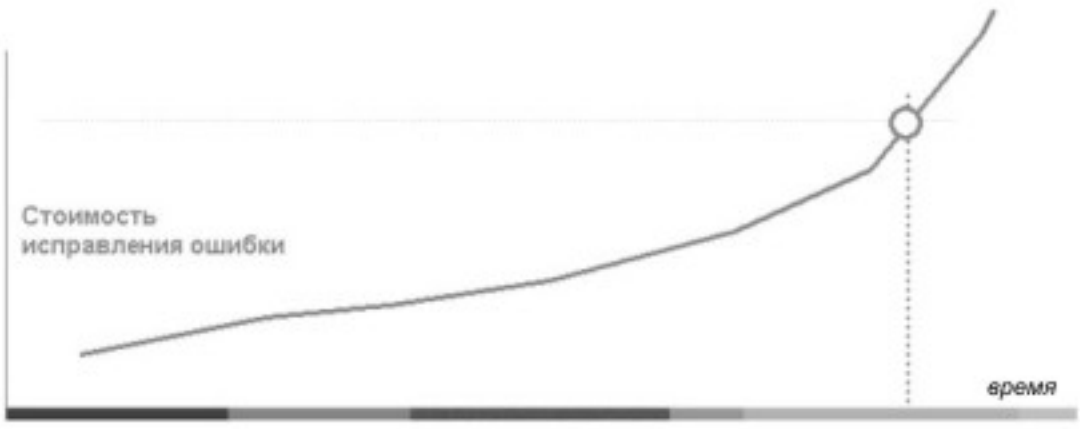

Рис. 48 Стоимость исправления ошибки при разработке "водопадом"

Чем позже разработчик обнаруживает ошибку, т.е. чем больше сделано по разработке программы, тем выше стоимость ее исправления (см. рис. 48). Причем, рост здесь близок к экспоненциальному. Также существуют и другие, не совсем очевидные минусы использования модели водопада. Например, сложно управлять рисками

некоторых типов (таких, как риски, связанные с использованием новых технологий или риски некорректного определения требований). Подобные риски могут проявить себя только на этапе реализации (при не достаточном тестировании), когда число возможных путей исправления ситуации намного меньше, чем в начале проекта.
Альтернативной методикой является “**Итеративная разработка**”[6].
Модель стала эволюционным развитием модели водопада. Процесс состоит из серии повторяющихся итераций (их число зависит от конкретного проекта), каждая из которых фактически является полноценным мини-проектом с фазами определения требований, анализа, верификации и т.д. В результате очередной итерации продукт приобретает новую функциональность или улучшения в существующей функциональности. Полный набор требований, зафиксированный границами проекта, оказывается реализованным после завершения финальной итерации.

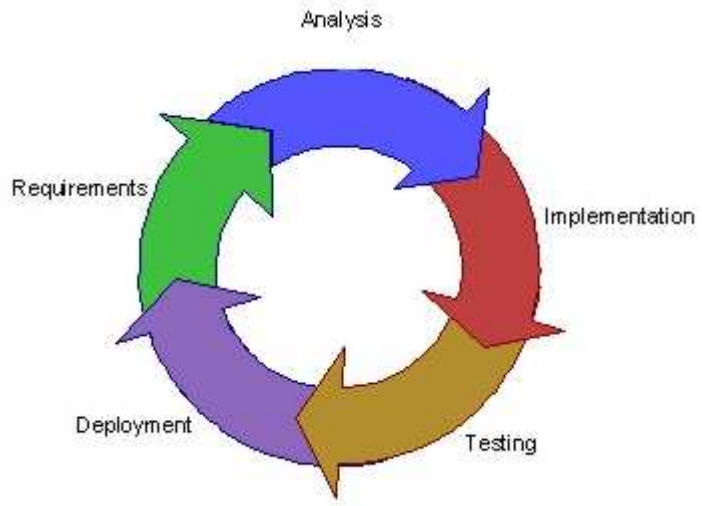

рис 49. Модель “Итеративная разработка”

На рис 49. requirements – требования, analysis – анализ и дизайн, implementation – реализация, testing – тестирование, deployment – развертывание, внедрение

В зависимости от требований разработки программного обеспечения, после каждой итерации разработчик может получить либо не дееспособный модуль приложения, либо узкоспециализированную, но рабочую программу.

Такая система разработки позволяет минимизировать всевозможные риски, связанные с внедрением новых технологий, изменением требований к проекту, просто неверным кодом, что часто бывает очень важно, так как приходится вносить изменения в программный код.

Если разработчик выбирает создание “Решателя” используя один из приведенных методов, то он получает следующие возможности:

1. Разработчик может внести изменения в ядро программы;
2. Разработчик может выполнить проверку вычислений на любом этапе, включая промежуточные, так как соответствующие средства создаются разработчиком,
3. Возможность создания удобной настройки программы и интерфейса, что не получиться сделать используя стороннюю программу.
4. Наличие возможности собственной разработки интерфейсов доступа к базе данных и выполнение формирования отчетности.
5. Если разработчик связан с программной системой и интерфейсами созданными некоторой компанией, то он не может вносить в нее изменения и выполнить полную проверку правильности результатов. Выбор будет определяться типом задачи, которая стоит перед разработчиком ППП.

6.2 Программные модули пакета программ принятие решений.

Пакет программ “Построение математических моделей” [5] позволяет по поставленной задаче (см. 3.1, рис. 3.6) либо ввести в “Решатель” готовую ММ из БД, либо вывести новую ММ. Ядро пакета позволяет строить математические модели следующих объектов:

1. механические объекты,
2. электрические объекты,
3. гидравлические объекты,
4. электромеханические, гидромеханические и др.
5. пневматические объекты.

Оконная форма пакета программ “Построение математических моделей” приведена на рис. 50. На этом рисунке приведена модель электромеханического авиационного прибора навигации (см. разделы В-2, рис. 1)

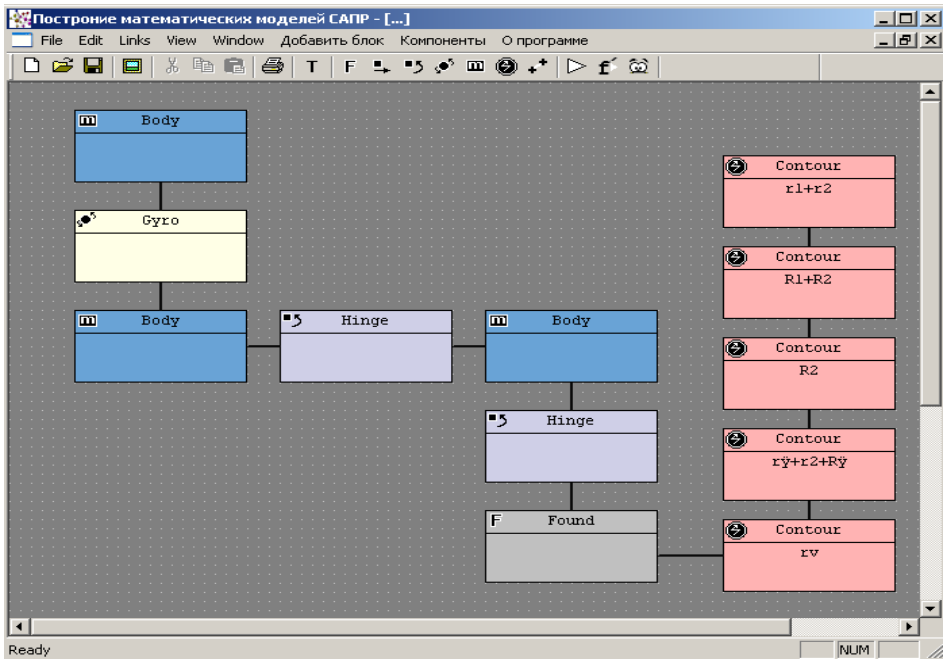

рис. 50 Оконная форма программы “Построение математических моделей”

Программные модули методам реализации ММ устройств в “Решателе” (см. 5.1,5.2) представлены на рис. 51 в виде связанной структуры с выводом оценочных характеристик для принятия проектных решений.

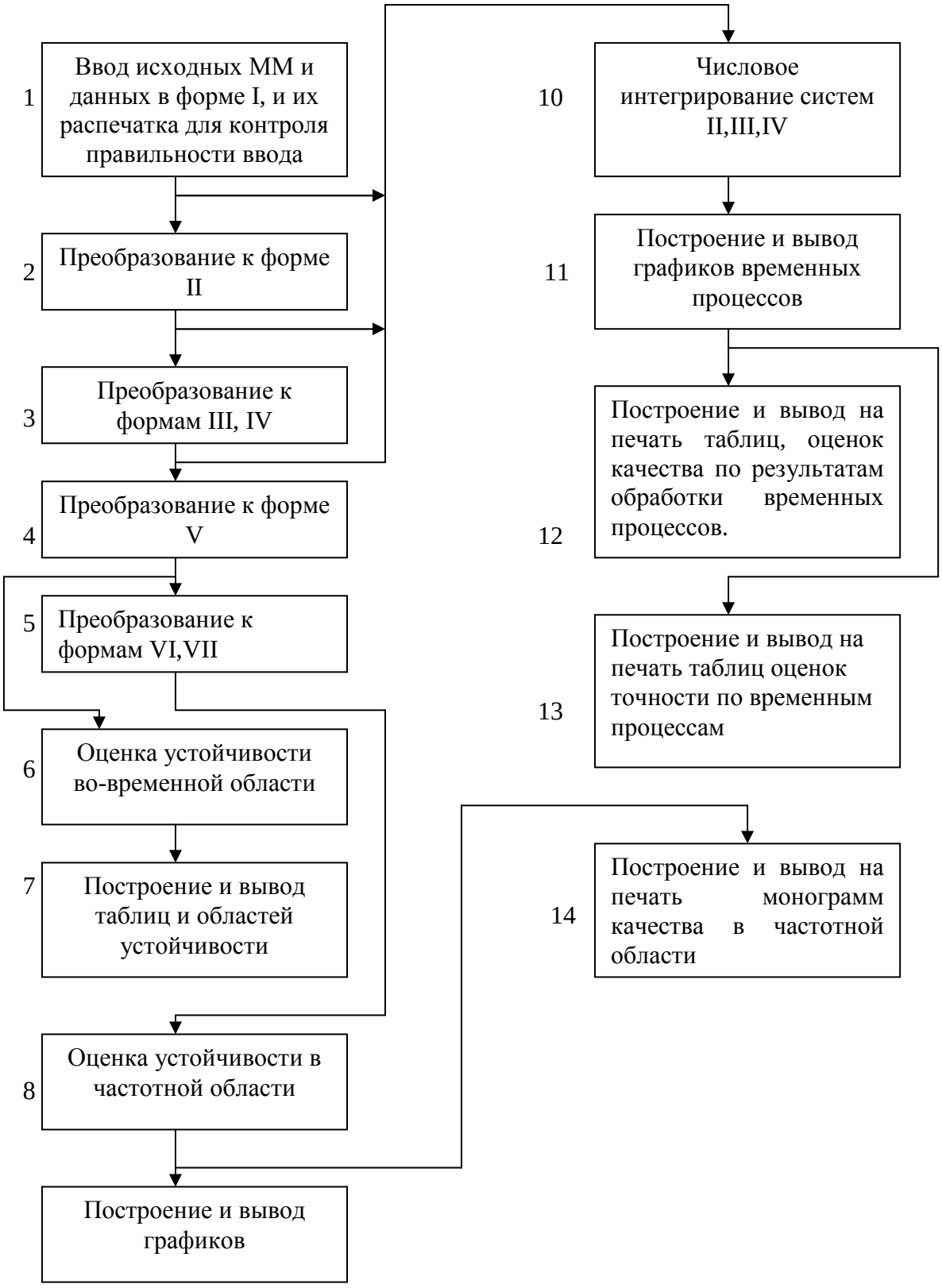

Рис. 51. Структура взаимодействия программных модулей

На рис. 51 введены обозначения ММ соответствующих форм (1), (5),(8),(10) приведенных в параграфе 2.5.2, а также (2) из 3.3 и другим формам ММ устройств и элементов.

I $F(Y^{(n)},Y^{(n-1)}\ldots,Y,\Lambda,X)=0,$

II $\dot{Y} = F(Y,\Lambda,X),$

III $\dot{Y} = AY + \varphi(Y,X),$

IV $\dot{Y} = AY + BX$,
V $\dot{Y} = AY$,
VI $a_0 p^n + a_1 p^{n-1} + \ldots + a_n = 0$,
VII $L(p)Y = R(p)X$,
VIII $Y = W(p)X$

Аналогично строятся и вводятся в “Решатель” структуры из программных модулей расчета элементов в соответствии с ММ, представленными в главе 2.6. Ввиду большого разнообразия таких расчетов (прочностных, жесткости, тепловых и др.) существующие расчетные ППП ANSYS, NASTRAN, MARC,ADAMS [9] встраиваются в “Решатель” в зависимости от области принятия проектных решений.

Алгоритм программы-монитора представлен в виде графа на рис. 52. Номера в вершинах графы соответствует номеру программного модуля в структуре рис. 51.

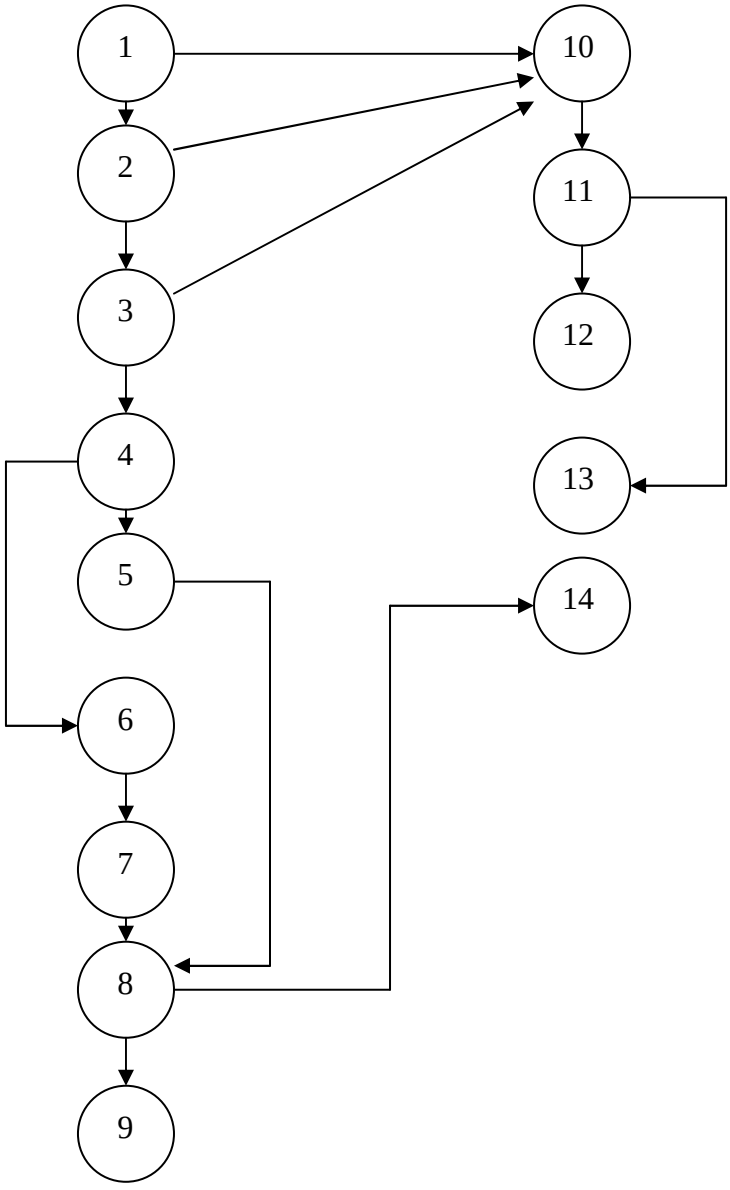

Рис. 52. Граф программы-монитора

Для ввода данного графа в “Решатель” используется матрица инциденции, соответствующая графу. Поскольку модули пронумерованы таким образом, что модуль с

меньшим номером передает данные модулю с большим номером, то матрица инциденции имеет треугольную форму. Эта матрица используется для управления программными модулями и пользователь, ЛПР, к ней доступа не имеет.

Программные модули по реализации машинно-аналитического метода (см. 5.3) строятся в соответствии с алгоритмом, приведенном на рис. 46. Наиболее подходящей системой, в которой реализуются представленные программные модули представляется Matlab-Simulink [7].

Программные модули по методам реализации ММ систем диалога в соответствии с методами принятия решений на основе матрицы полезности (см. 5.4) и эвристическо-аналитического метода (см. 5.5) реализуется на основе алгоритмов, приведенных в 5.4.1 (см. стр. 78) и (9) – (12) (см. стр. 82-83). Наиболее подходящей системой в этом случае представляется MathCAD.

ЛИТЕРАТУРА

1. Сольницев Р.И. Информационные технологии в проектировании: Учебн. Пособие Спб.: Изд-во ГУАП, 2007.
2. Сольницев Р.И., Гришанова Л.И., Слюсаренко А.С. Математическое обеспечение информационных технологий. Учебн. Пособие СПб.: Изд-во ГУАП, 2004.
3. Сольницев Р.И. Автоматизация проектирования систем автоматического управления. М.: Высшая школа, 1991,328 с.
4. Э. Мушик, П. Мюллер. Методы принятия технологических решений. М.: “Мир”, 1990
5. Сольницев Р.И., Майоров Н.Н. Построение математических моделей. Учебн. Пособие. СпбЭТУ ЛЭТИ, 2006
6. Ларнман К. Применение UML 2.0 и шаблонов проектирования: изд. 3-е, Изд-во: Диалектика-Вильямс, 2007, 736 с.
7. Черных И.В. Моделирование электротехнических устройств в MATLAB, SimPowerSystems и Simulink. 1-е издание, 2007 год, 288 стр.
8. ДС. В. Поршнев, И. В. Беленкова**.** Численные методы на базе Mathcad (+ CD). С-Пб: БХВ-Петербург, 2005, 456с.
9. Чигарев А.В., Кравчук А.С., Смалюк А.Ф. Ansys для инженеров. М: Машиностроение-1,2004.512с.