УДК 623.746

ФОРМИРОВАНИЕ МОДЕЛИРУЮЩЕЙ СРЕДЫ АВИАЦИОННОГО ТРЕНАЖЕРА

Б. К. Кемалов, Б. Ж. Куатов, Н. К. Юрков

Совокупность моделирующей среды и программного обеспечения авиационного тренажера (АТ) есть определенный аппаратно-программный продукт, выполняющий заданные функции по имитации самого летательного аппарата, его бортовых систем и окружающей среды в реальном масштабе времени.

Рациональное построение этого продукта влечет за собой повышение эффективности использования вычислительных средств, надежности авиационного тренажера, адекватности реализованных характеристик заданным, преемственности, скорости разработки, гибкости конфигурирования авиационных тренажеров разной степени сложности и применения [1].

Поэтому структура моделирующей среды и программного обеспечения авиационного тренажера с полным основанием может рассматриваться как объект самостоятельного исследования с целью ее дальнейшего совершенствования.

Структурный анализ сложной технической системы

Структурный синтез моделирующей среды авиационного тренажера проводится на основе комплексного анализа проблемы по следующим направлениям [2]:

– анализ дерева узлов вычислительной системы по их функциональному назначению;

– анализ информационных потоков между узлами и функциональными группами узлов в вычислительной системе;

– анализ соответствия выбранных средств вычислительной техники, реализующих узлы вычислительной системы, их функциональному соответствию;

– анализ выбора аппаратных средств сопряжения узлов в вычислительную систему;

– анализ соответствия характеристик работы отдельных узлов и всей вычислительной системы в целом заданным;

– анализ степени автономности узлов и функциональных групп узлов друг от друга;

– анализ надежности вычислительной системы;

– анализ структуры баз данных и средств поиска информации в них;

– анализ степени стандартизации и унификации, возможности преемственности;

– анализ гибкости и возможности реконфигурации;

– экономико-эргономический анализ.

Анализ дерева узлов моделирующей среды и информационных потоков в ней проведем на основе рассмотрения авиационного тренажера как объединения программно-аппаратных модулей, несущих четко определенные, унифицированные функции (модуль динамики полета, модули отдельных самолетных систем, модули навигации и т.п.) с регламентированным интерфейсом между ними.

В связи с частой сменой средств вычислительной техники моделирующая среда должна представлять собой реализацию логико-информационной схемы среды, которая определяется логико-функциональным подходом к сложным техническим системам, в том числе АТ [3].

Анализ соответствия выбранных вычислительных средств и средств сопряжения их функциональному назначению будем проводить с точки зрения их производительности, надежности и стоимости.

Анализ степени автономности узлов и их функциональных групп необходимо вести с учетом того, что каждый функциональный модуль моделирующей среды суть элемент специализированного или функционального тренажера. При автономной работе каждый модуль должен со-

здавать (генерировать) ту совокупность информации, которую он получает от смежных модулей, работая в комплексном режиме. Кроме того, автономность подразумевает возможность функционирования тренажера при отказе одного или нескольких модулей.

Анализ баз данных ведется с целью определения рациональности их организации для простоты и быстроты поиска информации в них.

Анализ степени стандартизации и унификации, возможности преемственности проводится с точки зрения обеспечения разработки и модернизации ряда авиационных тренажеров в сжатые сроки. Это должно проявляться при моделировании реальных систем летательного аппарата в выделении функционально-устойчивых ядер этих систем и тщательного моделирования этих ядер и конструктивного, часто меняющегося оформления.

Анализ гибкости и возможности реконфигурации должен проводиться с целью определения требуемых изменений в моделирующей среде и программном обеспечении при замене моделируемых систем или заданных характеристик. Во всяком случае, изменения внутри одного программно-аппаратного модуля не должны вести к изменениям других модулей. Кроме того, должна обеспечиваться возможность набора требуемой вычислительной системы из программно-аппаратных модулей разной степени сложности [4, 5].

Концепция модульности подразумевает свойство систем, в соответствии с которым отдельные ее элементы (модули) могут быть объединены, разделены и модифицированы без влияния на систему в целом и на другие модули.

Модульная структура авиационного тренажера

Модульный авиационный тренажер отличается от обычных современных АТ с «монолитной» структурой тем, что его функции разделены между несколькими четко определенными модулями, связанными друг с другом по унифицированной схеме. Эти модули должны выдавать и получать информацию, необходимую для работы тренажера. Каждый модуль может быть выполнен в различных вариантах, но интерфейсы между модулями должны быть стандартными.

При переходе на модульную концепцию западными фирмами, прежде всего, учитывается фактор «товарности» модуля, т.е. свойство модулей, согласно которому они могут создаваться и продаваться независимо друг от друга. В этом смысле «товарностью» уже обладают такие модули, как кабина, система подвижности, система визуализации, пульт инструктора. «Товарные» модули покупаются головной фирмой, производящей тренажеры, и объединяются на ней в законченное изделие (тренажер). Очевидно, что головная фирма заказывает и получает «товарные» модули для создания тренажера под конкретный вид ЛА.

Так как все без исключения модули обладают стандартным интерфейсом, головная фирма имеет возможность выбора и выбирает тот модуль (из предлагаемых изделий одного функционального назначения), который обладает лучшим соотношением «затраты/прибыль». Итак, одно из преимуществ модульности – независимость его создания, производства, т.е. «товарность». К преимуществам модульности относится также гибкость модуля, т.е. модуль может быть реорганизован для удовлетворения различным требованиям, например, введение дополнительных требований по реализации автономной тренировки, имитации дополнительных отказов.

Следующим преимуществом модульности является модифицируемость, т.е. модуль может быть изменен с целью отражения в нем изменений, связанных с реальным моделируемым объектом. При модификации модуля воздействие его на другие модули и систему в целом должно быть минимальным [6, 7].

В настоящее время сложно дать ответ на то, какой тренажер будет дешевле – модульный или с «монолитной» структурой. Потребуются дополнительные затраты для того, чтобы каждый создаваемый модуль отвечал поставленным требованиям. С другой стороны, объединение таких модулей, возможно, повлечет за собой быстроту исполнения и уменьшение стоимости.

При создании первого модульного тренажера невозможно добиться всех тех преимуществ, которые присущи модульному подходу. Эти преимущества заключаются в разработке тренажеров на базе существующих модулей. В этом случае экономия по времени создания и стоимости может быть большой.

При модульном подходе система, составленная из взаимосвязанных элементов, не может достигнуть полной модульности. Однако всегда можно получить некоторую степень модульности.

Модульный тренажер должен иметь модульную моделирующую среду и модульное программное обеспечение. Реально модуль вычислительной системы – это один или несколько функционально связанных ее узлов со стандартным логико-информационным интерфейсом со смежными модулями этой системы.

Чтобы определить модуль, т.е. определить его функции, необходимо определить входные связи, т.е. информацию, подлежащую обработке, выходные связи, частоту обмена информацией и т.п., другими словами, модуль должен быть определен функционально. Типы используемых ЭВМ, программное обеспечение не должны входить в определение модуля, это его конкретная реализация.

Должны быть полностью определены связи между модулями моделирующей среды, должны быть определены требования, определяющие разработку и внедрение модулей и элементов интерфейса [8, 9].

Эти понятия: определение, взаимосвязь и требования – являются главными в модульном подходе, и только полное их решение дает возможность успеха.

Переход к модульной структуре должен охватывать всю вычислительную систему.

Незавершенный, несогласованный модульный подход не дает никакого выигрыша, хотя и требует больших затрат на разработку по сравнению с «монолитным» подходом.

Наибольшая потенциальная опасность заключается в желании выбрать узкоспециальный подход, который жестко определяет архитектуру ЭВМ, жестко устанавливает структуру интерфейсов, диктует применение специального языка программирования.

Удачный модульный подход должен рассматривать все эти аспекты единым всеобщим образом. Он должен быть наиболее гибким и способным на некоторые исключения или отклонения без ущерба подхода в целом.

Поэтому модульный подход необходимо рассматривать на двух уровнях: логическом, на котором модули и интерфейс рассматриваются с функциональной стороны, и физическом, на котором моделирующая среда рассматривается как набор аппаратных и программных модулей. Основная цель данного рассмотрения – разделить эти уровни до такой степени, что изменение на физическом уровне не влечет за собой изменения на логическом уровне.

Классическими примерами логического уровня являются математическая модель системы решения навигационных параметров и математическая модель аэродинамики летательного аппарата. На этом уровне рассматриваются вопросы точности, объекты моделирования, входная и выходная информация модуля.

Языки программирования, характеристики ЭВМ, среда передачи информации не обсуждаются на логическом уровне.

Физический уровень – это среда, в которой функционируют логические модули и посредством которой они взаимодействуют. На этом уровне рассматриваются вопросы производительности аппаратных средств, их архитектура, системы программирования и собственно программы [10, 11].

Такое распределение позволяет достаточно гибко модернизировать модули на физическом уровне, не затрагивая логического. Можно использовать различные элементы физического уровня для создания конкретного тренажера.

Вычислительная система авиационного тренажера

Программы, лежащие в модуле вычислительной системы, сами представляют собой программный модуль. Этот модуль должен разрабатываться на логическом уровне и реализовываться на физическом. Программный модуль представляет собой некоторую иерархическую структуру более мелких программных модулей с той или иной степенью вложенности. Модульная программа – это логически структурированная программа, в которой любую часть логической структуры можно изменить, не вызывая изменений в остальных частях программы. Конечно, абсолютная независимость модулей недостижима. Они как-то связаны друг с другом. Нужно лишь стремиться, чтобы эти связи были минимальными или хотя бы четко оговоренными. Взаимосвязь или так называемое сцепление модулей классифицировано в табл. 1.

Таблица 1

Сцепление модулей ПО АТ

Системная характеристика модуля	Вид сцепления	Мера сцепления
1. Модуль не содержит о другом модуле никакой информации (слабое сцепление)	Независимое	0
2. Вызывающий модуль имеет имя вызываемого модуля, а также типы и значения некоторых его переменных	Данные	1
3. Параметры модуля содержат сведения о внутренней структуре данных	Образец	3
4. Модули разделяют одну и ту же глобальную структуру данных	Общая область	4
5. Один модуль имеет данные о внутренних функциях другого и управляет решениями внутри другого с помощью передач флагов, переключателей или кодов, предназначенных для выполнения функций управления	Управление	5
6. Модуль имеет доступ к данным в другом модуле через внешнюю точку входа	Внешние ссылки	5
7. Модули кодов команд, которые перемежаются друг с другом (сильное сцепление)	Коды	7

Когда говорится о независимости программного модуля, имеется в виду, что один или несколько определенных программных факторов тоже будут независимыми, например [12]:

– логическая структура программы;
– аргументы или параметры модуля;
– внутренние переменные, таблицы и константы;
– структура и формат базы данных;
– модульная структура управления программой.

Предполагая эти факторы неизменными, можно требовать независимость отдельных модулей программы. Если интерфейсы между модулями определены, то в этом случае можно заменять модуль функционально ему эквивалентным, не вызывая при этом никаких последствий ни в каком другом модуле программы.

Нет единых законов разделения программы на модули. Их можно делать более мелкими, например, требуя, чтобы объем модуля был не более 50 операторов языка высокого уровня, или более крупными, объединять несколько функций в один модуль, из модуля обращаться к другим модулям и т.д. Характеристикой модуля по его внутреннему содержанию является связность. По связности модули ПО АТ классифицированы в табл. 2.

Таблица 2

Связность модулей ПО АТ

Системная характеристика модуля	Вид связности	Мера связности
1. Выполняет единственную функцию. Модуль типа вход – преобразователь – выход. Не может быть разбит на несколько других модулей, имеющих связность такого же уровня	Функциональная (сильная связь)	10
2. Модуль может быть разбит на части, выполняющие независимые функции, которые совместно реализуют единственную функцию (оценка и обработка данных)	Информационная (последовательная)	9
3 Модуль, состоящий из независимых модулей, разделяющих структуру данных. Общая структура данных является основой его организации. Запоминание и поиск данных	Коммуникативная	7
4. Модуль последовательно выполняет набор функций, непосредственно относящихся к процедуре решения задачи	Процедурная	5
5. Модуль содержит части, функционально не связанные в один и тот же момент обработки	Временная (по классу)	3
6. Модуль объединяет операторы по принципу их функционального подобия, а для его настройки используется алгоритм переключения	Логическая	1
7. Объединение в модуль абсолютно не связанных частей ПО	По совпадению	0

Достоинствами модульности в программировании являются:

а) модульные программы легко составлять и отлаживать. Функциональные компоненты такой программы могут быть написаны и отлажены порознь;

б) модульную программу легче сопровождать и модифицировать. Функциональные компоненты могут быть изменены, переписаны или заменены без изменений в остальных частях;

в) разработкой модульной программы легче управлять.

К недостаткам модульного подхода в программировании можно отнести:

а) модульность часто требует большой дополнительной работы;

б) модульный подход часто требует дополнительных затрат времени центрального процессора;

в) модульный подход часто требует дополнительные ресурсы оперативной памяти.

Но эти недостатки часто переходят в достоинства из-за преемственности модулей, возможности их тщательной разработки, применения библиотечных программ, минимизированных по времени выполнения или памяти стандартных процедур и функций [13, 14].

Концепция модульности моделирующих сред авиационных тренажеров требует разработки формальных методов выделения отдельных модулей, как правило, основанных на теории графов и теории цепей.

Распределение программных модулей по узлам моделирующей среды

Требуется распределить N взаимодействующих программных модулей по M узлам моделирующей среды с минимальным количеством параметров обмена между ними. При этом суммарное время выполнения программных модулей в каждом узле не должно превышать некоторое T_i.

Количество параметров обмена между отдельными программными модулями задается матрицей смежности A размерностью $(n \times n)$, время выполнения программных модулей вектором $B(n)$.

Для построения формальной математической модели распределения удобно использовать теорию графов. При этом пакет программных модулей интерпретируется как ненаправленный мультиграф (т.е. граф у которого существует хотя бы одна пара вершин, соединенных m ребрами), в котором каждому программному модулю соответствует вершина мультиграфа, а параметрам обмена – его ребра.

Тогда задача распределения программных модулей по узлам вычислительной сети формулируется следующим образом.

Задан мультиграф $G(X,U)$. Требуется «разрезать» его на отдельные подграфы так, чтобы число ребер, соединяющих эти подграфы, было минимальным, т.е. минимизировать

$$\sum_{i=1(i\neq j)}^{k}\sum_{j=1}^{k} U_{ij} \quad \text{для } \forall G_i(X_iU_i) \subset G(X,U) \ .$$

Из $G_i(X_i,U_i) \neq G_j(X_j,U_J)$ следует, что $X_i \bigcap X_j = 0$, $U_i \bigcap U_{ij} = 0$, где U_{ij} – множество ребер, соединяющих подграфы $G_i(X_i,U_i)$ и $G_j(X_j,U_j)$. Ограничением в рассматриваемой задаче является суммарное время выполнения программных модулей для каждого узла.

Известные алгоритмы распределения можно условно разбить на пять групп: алгоритмы, использующие методы целочисленного программирования; последовательные алгоритмы; итерационные алгоритмы; смешанные алгоритмы; алгоритмы, основанные на методе ветвей и границ.

Алгоритмы первой группы хотя и позволяют получить точное решение задачи, однако для мультиграфа реальной сложности практически не реализуемы на ЭВМ.

В отечественном тренажеростроении наибольшее распространение получили приближенные алгоритмы распределения (последовательные, итерационные, смешанные).

При использовании последовательных алгоритмов сначала по определенным правилам выбирают первую вершину графа, затем осуществляют последовательный выбор вершин (из числа нераспределенных) и присоединение их к формируемому узлу графа. После образования первого узла переходят ко второму и т.д. до получения желаемого разрезания исходного графа.

В итерационных алгоритмах начальное разрезание графа на куски выполняется произвольным образом: оптимизация компоновки достигается парными или групповыми перестановками

вершин графа из различных кусков. Процесс перераспределения вершин заканчивают при получении локального экстремума целевой функции, удовлетворяющего требованиям разработчика.

В смешанных алгоритмах распределения для получения начального варианта «разрезания» используется алгоритм последовательного формирования кусков; дальнейшая оптимизация решения осуществляется перераспределением вершин между отдельными кусками графа.

В последовательных алгоритмах распределения «разрезание» исходного графа $G(X, U)$ на модули $G_i(X_i, U_i), (i+1, ..., k)$ сводится к следующему.

В графе $G(X,U)$ находят вершину $X_i \in X_i$ принадлежащую X с минимальной локальной степенью $p(X_i) = \min p(X_f)$, для всех X_f, принадлежащих X, где $p(X_f)$ равен сумме A_{fp} (p=1,...,N).

Если таких вершин несколько, то предпочтение отдается вершине с максимальным числом кратных ребер. Из подмножества вершин, смежных с вершинами формируемого куска графа $G_1(X_1, U_1)$, выбирают ту, которая обеспечивает минимальное приращение связей куска с еще нераспределенными вершинами $d(X_j) = \min d(X_r)$ (для всех X_r принадлежащих X_s), где $d(X_r) = p(X_r) - 2$ (сумма A_{re} принадлежит E).

Данную вершину X_j, принадлежащую X/X_1, включают в $G_1(X_1,U_1)$, если не происходит нарушение ограничения по сумме времени выполнения программных модулей узла, т.е. если X_j принадлежит ΓX_1 & $d(X_j) = p(X_j) - 2$ (сумма A_{je} принадлежит E) = $\min d(X_r)$ (X_r принадлежит ΓX_1) & сумма TIME(X_g) (g принадлежит объединению j и E и не больше m], тогда X_j включается в X_1, где A_{je} – элемент матрицы смежности исходного графа $G(X,U)$; $d(X_g)$ – относительный вес вершины X_g, равный приращению числа внешних ребер куска $G_1(X_1, U_1)$ при включении вершины X_g во множество X_1; E – множество индексов вершин, включенных в формируемый кусок графа на предыдущих шагах алгоритма; m – максимально допустимое суммарное время выполнения программ для отдельно взятого узла.

Указанный процесс продолжается до тех пор, пока присоединение очередной нераспределенной вершины X_j к узлу $G_1(X_1,U_1)$ не приведет к нарушению ограничения по времени выполнения программных модулей для данного узла.

После образования первого узла процесс повторяется для формирования второго, третьего и т.д. с той лишь разницей, что рассмотрению подлежат только вершины, не вошедшие в предыдущие узлы.

Достоинством последовательных алгоритмов является простота их реализации на ЭВМ и высокое быстродействие.

Недостатком этих алгоритмов является получение результатов, которые в общем случае могут быть далекими от оптимальных, что сужает область их использования.

Заключение

Таким образом, представлено рациональное построение моделирующей среды и программного обеспечения авиационного тренажера. Обоснована необходимость комплексного анализа проблемы структурного синтеза сложной технической системы, построенной на основе модульного принципа конструирования.

Список литература

1. Юрков, Н. К. Синтез системы управления интеллектуальной компьютерной обучающей системой / Н. К. Юрков, А. В.Затылкин, Б. К. Кемалов // Новые промышленные технологии. – 2011. – № 2. – С. 58–61.
2. Юрков, Н. К. Синтез автоматизированной системы оценивания качества пилотирования на авиационном тренажере / Н. К. Юрков, А. И. Годунов, Ю. Г. Квятковский // Известия высших учебных заведений. Поволжский регион. Технические науки. – 2012. – № 1 (21). – С. 58–64.
3. Юрков, Н. К. Синтез управления движением мобильного робота по траектории методом интеллектуальной эволюции / Н. К. Юрков, А. И. Дивеев, Е. Ю. Шмалько // Труды международного симпозиума Надежность и качество, 2013. – Т. 1. – С. 188–190.
4. Принципы создания сквозных управляемых динамических систем применительно к авиационным тренажерам / Н. К. Юрков, А. М. Данилов, Э. В. Лапшин, И. А. Гарькина // Информационные технологии в проектировании и производстве. – 2004. – № 2. – С. 53–57.

5. Юрков, Н. К. Проблемы развития авиационного тренажеростроения / Н. К. Юрков, Э. В. Лапшин, А. В. Блинов // Информационные технологии в проектировании и производстве. – 1999. – № 3. – С. 33–39.
6. Юрков, Н. К. Информационные модели проектирования интеллектуальных тренажеров широкого профиля / Н. К. Юрков, Э. В. Лапшин, А. В. Блинов // Измерительная техника. – 2000. – № 8. – С. 23–27.
7. Принципы создания сквозных управляемых динамических систем применительно к авиационным тренажерам / Н. К. Юрков, А. М. Данилов, Э. В. Лапшин, И. А. Гарькина // Информационные технологии в проектировании и производстве. – 2004. – № 2. – С. 53–57.
8. Авиационные тренажеры модульной архитектуры : моногр. / Э. В. Лапшин, А. М. Данилов, И. А. Гарькина, Б. В. Клюев, Н. К. Юрков. – Пенза : Информационно-издательский центр ПГУ, 2005. – 148 с.
9. Юрков, Н. К. Состояние и перспективы развития авиационного тренажеростроения / Н. К. Юрков, Э. В. Лапшин // Информационные технологии в образовании, науке и производстве : сб. тр. 2-й Междунар. науч.-практ. конф. (Серпухов, 30 июня – 4 июля 2008 г.). – Серпухов, 2008. – С. 554–565.
10. Лысенко, А. В. Анализ особенностей применения современных активных систем виброзащиты для нестационарных РЭС / А. В. Лысенко, Г. В. Таньков, Д. А. Рындин // Труды международного симпозиума Надежность и качество, 2013. – Т. 2. – С. 155–158.
11. Юрков, Н. К. Моделирование прогнозно-оптимизационной деятельности оператора авиационного тренажера / Н. К. Юрков, Б. К. Кемалов // Computer – based conference : тр. Междунар. науч.-техн. конф. – Пенза : Пензенская государственная технологическая академия, 2011. – Вып. 14. – С. 102–108.
12. Информационная технология многофакторного обеспечения надежности сложных электронных систем / Н. К. Юрков, А. В. Затылкин, С. Н. Полесский, И. А. Иванов, А. В. Лысенко // Надежность и качество сложных систем. – 2013. – № 4. – С. 75–79.
13. Юрков, Н. К. Обеспечение комплексной адекватности авиационных тренажеров / Н. К. Юрков, А. И. Годунов, Б. К. Кемалов // Известия высших учебных заведений. Поволжский регион. Технические науки. – 2011. – № 3 (19). – С. 15–24.
14. Юрков, Н. К. Синтез автоматизированной системы оценивания качества пилотирования на авиационном тренажере / Н. К. Юрков, А. И. Годунов, Ю. Г. Квятковский // Известия высших учебных заведений. Поволжский регион. Технические науки. – 2012. – № 1 (21). – С. 58–64.

Кемалов Берик Каирович
кандидат технических наук,
заместитель начальника департамента образования и науки министерства обороны Республики Казахстан,
(010000, Казахстан, г. Астана, пр. Достык, 14)
E-mail: kemalov.bk@gmail.com

Куатов Бауржан Жолдыбаевич
заместитель начальника по учебной и научной работам,
Военный институт Сил воздушной обороны Республики Казахстан им. Т. Я. Бегельдинова
(463024, Казахстан, г. Актобе, пр. А. Молдагуловой, 16)
8-(713)-270-37-82
E-mail: kuatov.baurjan@mail.ru

Юрков Николай Кондратьевич
доктор технических наук, профессор,
заведующий кафедрой конструирования и производства радиоаппаратуры,
Пензенский государственный университет
(440026, Россия, г. Пенза, ул. Красная, 40)
8-(412)-56-43-46
E-mail: yurkov_NK@mail.ru

Kemalov Berik Kairovich
candidate of technical sciences,
Deputy Head of the Department of education and science of the Ministry of defence of the Republic of Kazakhstan,
(010000, 14, Dostyk ave., Astana, Kazakhstan)

Kuatov Baurzhan Zholdybaevich
deputy chief of academic and scientific works,
Military Institute of Air Defense Forces of the Republic of Kazakhstan named after T. Ya. Begel'dinova
(463024, 16 A. Moldagulova avenue, Aktobe, Kazakhstan)

Yurkov Nikolay Kondrat'evich
doctor of technical sciences, professor,
head of sub-department of radio equipment design and production,
Penza State University
(440026, 40 Krasnaya street, Penza, Russia)

Аннотация. Рациональное построение моделирующей среды и программного обеспечения авиационного тренажера является актуальной проблемой. Обосновывается необходимость комплексного анализа про-

Abstract. Efficient simulation environment and aviation training device software is the actual problem. The necessity of a comprehensive analysis of the problem of structural synthesis of a complex tech-

блемы структурного синтеза сложной технической системы, построенной на основе модульного принципа конструирования.

Ключевые слова: авиационный тренажер, моделирующая среда, анализ модульной структуры, структурный синтез, моделирование, вычислительная система, теория графов.

nical system built on the basis of a modular design principle.

Key words: aviation simulator, simulation environment, analysis of modular structure, structural synthesis, modeling, computational system, graph theory.

УДК 623.746

Кемалов, Б. К.

К проблеме структурного синтеза моделирующей среды авиационного тренажера / Б. К. Кемалов, Б. Ж. Куатов, Н. К. Юрков // Надежность и качество сложных систем. – 2015. – № 1 (9). – С. 9–16.