УДК 681.51:629.78

ТЕХНОЛОГИЯ ПРОЕКТИРОВАНИЯ НАДЕЖНЫХ УПРАВЛЯЮЩИХ АЛГОРИТМОВ РЕАЛЬНОГО ВРЕМЕНИ ДЛЯ КОСМИЧЕСКИХ АППАРАТОВ

© 2004 А. А. Тюгашев

Самарский государственный аэрокосмический университет

Рассматривается проблема создания надежных алгоритмов управления реального времени, устанавливаемых на борт космического аппарата. Описывается базирующаяся на модели семантики алгоритма методология проектирования управляющих алгоритмов, обеспечивающая повышение надежности. Приводятся сведения об автоматизированной системе проектирования, поддерживающей данную технологию.

Одним из наиболее сложных компонентов технического комплекса, которым является космический аппарат (КА), является бортовой комплекс программ [1].

Широко применяемой и апробированной методикой повышения надежности программ является тестирование.

Однако проверить действительно ***все*** возможные варианты исполнения при достаточно сложной схеме программы и количестве логических условий в ней, равно как и правильность исполнения программы на ***всех*** возможных комбинациях исходных данных (ИД), практически невозможно. Таким образом, даже полностью успешное тестирование на некотором подмножестве вариантов исполнения управляющего алгоритма (УА) на некотором подмножестве возможных исходных данных (ИД) не дает гарантии правильности программы. Более того, в случае реального времени проблема усложняется тем фактором, что алгоритм взаимодействует с непредсказуемой внешней средой, изменения в которой могут происходить в произвольные моменты времени, и необходимо вводить время в качестве важной составляющей набора исходных данных.

В настоящей работе ставится задача построения методики повышения надежности программы при ее проектировании.

Сформулируем теоретический базис предлагаемой технологии.

Когда проводят исследование обычного алгоритма (либо программы), то, как правило, его ценность определяется результатом, который получается по окончании его работы. Под результатом подразумевается некоторый набор выходных данных, полученных в результате обработки набора ИД. Состояния программы (алгоритма), как начальное и конечное, так и промежуточные, определяются совокупностью значений в программной памяти. Например, известна общая формализация понятия программы, исходящая из формулировки так называемых постусловий и предусловий [2]. Данные условия есть часть языка *алгоритмических логик*, включающих условия вида

$$\{U\}\ S\ \{B\}, \qquad (1)$$

читающиеся следующим образом: "U – условие, относящееся к исходным данным программы S, истинное до ее выполнения, B – условие, относящееся к выходным данным программы S, которое должно быть истинным после ее исполнения".

Данный подход не может быть непосредственно применен для случая управляющих алгоритмов реального времени (УА РВ), поскольку для них важным условием правильности является осуществление корректного управления бортовой аппаратурой на всем промежутке времени активного существования КА. Более того, для управляющей программы КА неприменим классический подход к алгоритму как к набору действий для получения ***по завершению*** его работы определенного результата.

С точки зрения правильности работы УА РВ его надежность может быть определена как успешное исполнение КА его целевой задачи при любом возможном развитии ситуации, или в обозначениях пред- и постусловий:

$$\{ U(D_0,t_0) \}\ YS\ \{ B(D_k(t_1,t_2,\ldots,t_k),t_k)\}. \quad (2)$$

В момент времени t_0 начала функционирования КА истинно условие корректного задания исходных данных D_0, а к моменту t_k завершения работы управляющего алгоритма *YS* истинно условие D_k, означающее успешное выполнение всех целевых задач на всех заданных моментах времени $t_1, t_2, ..., t_k$ на всем промежутке времени функционирования КА.

Отсюда логически следует предложенное автором [3] представление семантики УА РВ в виде набора четверок объектов:

$$\text{УА РВ} = \{< f_i, t_i, \tau_i, \overline{l_i} >\}, i = \overline{1, N}, \quad (3)$$

где f_i – функциональная программа (действие); t_i - момент начала исполнения действия; τ_i - длительность действия; $\overline{l_i}$ - логический вектор, обуславливающий действие.

Таким образом, управляющий алгоритм должен обеспечивать на некотором непустом множестве временных меток (“включений”) выполнение определенных действий по управлению КА, зависящих от текущей ситуации на борту, отражаемой вектором значений логических переменных. В этом заключается целевая задача, решение которой обеспечивается алгоритмом [4].

УА РВ, отвечающие за качественное выполнение бортовой аппаратурой КА своих функций, являются сложной технической системой, а их проектирование – сложной инженерно-технической задачей.

Основная идея технологии проектирования надежных алгоритмов управления ГРАФКОНТ заключается в постепенном конструировании сложного УА из априори надежных блоков – более простых программ управления «базового» уровня – так называемых функциональных программ (ФП) [5].

Если при этом обеспечивается правильность алгоритма конструирования, т. е. соединения ФП в единое целое, то можно обеспечить повышение надежности получаемого в результате управляющего алгоритма.

Основой принципов надежного соединения функциональных программ, адекватно описывающего все возможные варианты в рамках логики и внутренних временных соотношений управляющего алгоритма, в технологии ГРАФКОНТ являются исчисление УА РВ и операции алгебры УА [6].

К функциональным программам и получаемым промежуточным результатам их объединения по правилам алгебры УА применяются операции, показанные на рис. 1.

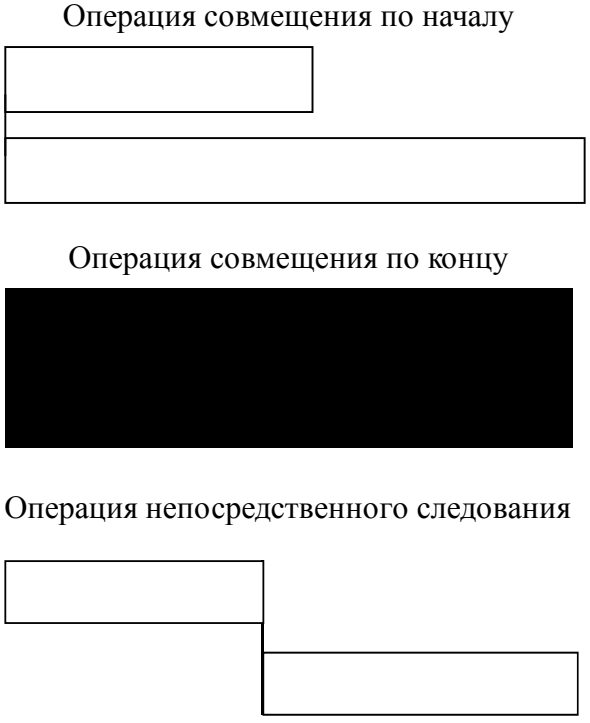

Рис. 1. Операции над УА

Данная методика позволяет описать все возможные варианты сочетаний моментов запуска ФП по времени и все возможные варианты исполнения УА РВ.

Для того, чтобы организация бортовой вычислительной системы (БВС) позволяла выполнять программы, реализующие УА, описываемые приведенной алгебраической моделью, необходима реализация в БВС нескольких принципов.

Бортовая операционная система (БОС), управляющая ходом вычислительного процесса в БВС, должна являться мультипрограммной и прерываемой, то есть диспетчер должен иметь возможность прервать выполнение любого процесса и предоставить вычислительный ресурс той задаче, которой он более необходим. Отсюда следует необходимость существования приоритетов задач.

При этом соответствие вышеприведенным требованиям могут обеспечить различные дисциплины организации вычислительного процесса в БВС [4]:

1. Регулярная синхронная с выделением задачам квантов времени процессора в соответствии с некоторым заранее составленным расписанием.

2. Приоритетная асинхронная, когда каждая задача решается по заявкам на ее выполнение, которые вырабатываются асинхронно текущему вычислительному процессу как со стороны внешних событий, так и самим бортовым программным обеспечением (БПО). В данном случае формируется специальная очередь готовых к исполнению задач, ждущих выделения вычислительного ресурса специальной программой БОС – планировщиком.

При выборе принципов организации вычислительного процесса в БВС необходимо учитывать, что БВС системы управления КА предназначена для решения фиксированного круга задач. Поэтому загрузка управляющей БВС детерминирована и может быть запланирована заранее путем составления расписания, учитывающего временную диаграмму работы системы управления КА.

По ряду причин, включающих, в частности, простоту коррекции БПО и добавление в него дополнительных задач, возможность оперативного дистанционного изменения состава решаемых БПО задач, более оптимальную загрузку вычислительных ресурсов для сложных многофункциональных комплексов БПО, в которых моменты начала и окончания решения задач могут меняться в широких пределах в зависимости от временных разбросов работы бортовой аппаратуры (БА) и исходных данных, передаваемых с Земли, предпочтительно использование приоритетной динамической асинхронной организации вычислительного процесса [4].

Ключевыми параметрами качества функционирования БВС и БПО как элемента системы управления КА при этом являются:

1. $T_{сист}$ – промежуток времени от поступления заявки на выполнение задачи в БВС до момента окончания ее обслуживания, включая время ожидания задачей дообслуживания в случае, если задача была прервана другой задачей в процессе выполнения («задержка»).

2. Загрузка ρ бортовой вычислительной системы, определяемая отношением времени работы без простоя БВС на определенном временном интервале к величине этого интервала. При этом необходимо обеспечить исполнение в реальном времени не менее 50 программ в любой последовательности с задержками, не превышающими допустимые. Для каждой конкретной программы управления величина допустимой задержки определяется с учетом функциональной задачи, решаемой управляемой данной программой БА, и может колебаться от единиц и десятков миллисекунд до десятков секунд [4].

Согласно теории массового обслуживания загрузка ρ определяется по формуле

$$\rho = \lambda / \mu, \qquad (4)$$

где λ - интенсивность поступления заявок на обслуживание, $с^{-1}$; μ - интенсивность обслуживания заявок, $с^{-1}$.

Интервалы времени между последовательными заявками на исполнение для простейшего входного потока (стационарного, без последействия, ординарного) – независимые случайные величины с функцией распределения

$$P(\tau) = 1 - e^{-\lambda \tau}, \quad (5)$$

где τ – интервал времени между заявками, с.

Для простейшего входного потока и экспоненциального распределения длительности обслуживания среднее время задержки определяется как

$$T_{сист} = 1 / \mu(1 - \rho). \quad (6)$$

Однако использование аналитических методов теории массового обслуживания в данном случае приводит к необходимости учета множества различных деталей организации вычислительного процесса в БВС и в конечном итоге к необходимости численных расчетов по громоздким математическим моделям [4].

При значении загрузки БВС $\rho > 0{,}8$ резко возрастают задержки на исполнение заявок в БВС на решение задач, и поэтому целесообразно ограничивать загрузку БВС значениями до $\rho = 0{,}8$.

В связи с этим особую важность представляет возможность автоматизированного получения временных характеристик конструктивным путем по результатам проектирования управляющих программ комплексного функционирования БА в рамках технологии ГРАФКОНТ.

При проектировании конкретного комплекса БПО работа начинается с анализа типовых временных диаграмм совместной работы программ БВС при реализации каждой из основных функциональных задач КА, которым соответствует специальная разновидность программ управления – программы комплексного функционирования и которые приводятся в проектном документе – циклограмме наложений программ БПО.

Циклограммы разбиваются на участки, характеризующиеся постоянством состава программ, могущих работать одновременно. Количество циклограмм определяется числом основных режимов, в которых функционирует бортовая аппаратура КА (ориентация, маневр и др.).

Часто требуется, чтобы некоторые из задач, решаемых БВС, имели меньшее время ожидания, а соответствующие им заявки – меньшее время задержки по сравнению с задачами других типов. Например, аварийные сигналы, приходящие по системе прерывания БВС, требуют практически немедленной реакции по включению систем аварийной защиты. Программы управления высокодинамичными процессами, например угловой стабилизацией, требуют меньшего времени задержки, чем программы управления менее динамичными процессами, например включения аппаратуры автономной навигации [4].

Исходными данными для распределения приоритетов являются циклограммы наложений и временные характеристики программ, включающие в себя времена выполнения отдельных участков программ и допустимые задержки на их выполнение. Допустимая задержка на работу участков программ определяется разработчиками функциональных программ (ФП), исходя из логики работы управляемой БА.

Вначале определяется загрузка БВС на каждом участке циклограммы:

$$\rho_j = \sum_{i=1}^{m_j} \frac{t_i}{T_j}, \quad (7)$$

где t_i – время исполнения i-ой программы; T_j – период времени j-го участка; m_j – количество программ, работающих одновременно на j-м участке циклограммы

Затем проверяется, не превышает ли значение загрузки 80 %, и в противном случае предпринимаются меры для ее снижения путем изменения количества программ, работающих на данном j-м участке. Распределение приоритетов является итерационным процессом. Первоначально приоритеты назначаются исходя из условия, что участкам с большим временем выполнения и большими допустимыми задержками назначаются более низкие приоритеты. Если задержка какой-либо из программ, определяемая суммированием времени исполнения более приоритетных программ и времени работы БОС, превышает допустимую, то производится перераспределение приоритетов: повышение приоритета участков, для которых получено превышение допустимых задержек, и снижение приоритетов участков, для которых присутствует резерв по задержкам. Данный числен-

ный подход к распределению приоритетов, оценке загрузки БВС и определению значений допустимых задержек производится специально разработанным пакетом программ [4].

Задача построения (генерации) управляющей программы, реализующей управляющий алгоритм с заданной семантикой, сводится к построению логико-временной схемы функционирования, то есть фактически в терминах теории схем программ – схемы программы, но с учетом аспекта реального времени.

Будем называть такую схему ***многовходовой моделью*** управляющего алгоритма.

Таким образом, задача генерации ПКФ сводится к построению отображения

S => MWM,

где S – семантика УА в виде набора четверок; MWM – многовходовая модель УА.

Многовходовая модель представляет собой набор входов (включений) УА в определенные моменты времени. При этом считаем, что длительность входа равна нулю в условной временной шкале исполнения управляющей программы, т. е. длительность операций – функциональных задач внутри входа стремится к нулю по шкале бортового времени и несоизмерима с интервалами времени между входами, например, миллисекунды и секунды. Этот факт приводит к тому, что внутри входа генератор может произвольно переставлять моменты запуска f_i, и при необходимости сохранения определенного порядка последовательность надо задавать дополнительно в виде информации о существовании отношения предшествования на множестве f_i функциональных задач, обуславливаемого, например, тем, что одна программа использует данные, формируемые другой, то есть между ними существует информационная связь.

В наиболее простом случае число включений равно просто количеству различающихся друг от друга моментов времени t_i, содержащихся в семантике УА и соответствующих моментам запуска входящих в управляющий алгоритм отдельных функциональных программ f_i.

Соединенная передачами управления между входами совокупность входов образует граф

$$MWM = < W, U >, \quad (8)$$

где W – множество вершин (входов); U – множество дуг (передач управления между входами), то есть бинарное отношение WxW.

В общем случае, при наличии циклических действий в УВ граф может содержать циклы. В случае наиболее распространенного ациклического варианта должно выполняться условие последовательного запуска при передаче управления: если существует дуга (W_i, W_j), то $t_{0Wi} < t_{0Wj}$, где t_{0Wi} - момент включения входа W_i, t_{0Wj} - момент включения входа W_j.

Каждый вход представляет собой, в свою очередь, логическую схему (последовательность проверок условий и действий f_i), реализующих логику входа.

Вход можно рассматривать как совокупность линейных участков и ветвлений по результатам проверки логических условий [2].

Линейным участком будем называть последовательность функциональных задач (ФЗ) без проверок логических условий и передач управления. Линейный участок может включать несколько f_i или ни одной.

Тогда логическая схема входа будет представлять собой граф в виде дерева:

$$W = < LU, Y >, \quad (9)$$

где LU – множество вершин графа (линейных участков); Y – множество дуг - передач управления (ветвлений).

Каждая вершина логической схемы входа (помимо листьев, соответствующих заключительным линейным участкам входа) будет иметь две исходящих из нее дуги, соответствующие вариантам выполнения и невыполнения логического условия, то есть дерево будет бинарным. Пример графа входа приведен на рис. 2.

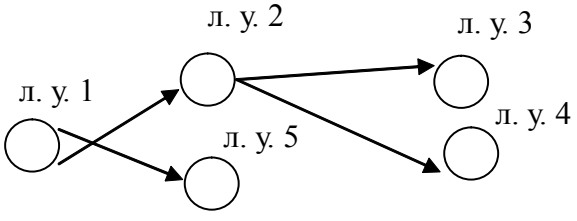

Рис. 2 Логическая схема входа

Ясно, что если внутри линейного участка важна последовательность выполнения ФЗ f_i и f_j, то должно выполняться условие $t_i = t_j$. Итак, управляющий алгоритм в виде набора действий, которые следует предпринять в заданные моменты времени в зависимости от заданных логических условий, необходимо реализовать в контексте конкретной операционной системы БВС реального времени.

Логические условия, фигурирующие в многовходовой модели, по времени их значимости (актуальности) могут относиться к одному из трех типов [2]:

1. $\alpha(t)$, то есть интересует значение логической переменной в текущий момент времени t, и с течением времени значение α может меняться;

2. $\alpha(t_0)$, когда значение переменной может меняться, но интересует ее значение в указанный момент времени t_0 (часто это момент начала исполнения УА), а изменения значения во времени не должны влиять на исполнение ПКФ;

3. $\alpha_{стат}$, когда значение логической переменной не претерпевает изменений в процессе исполнения УА.

Очевидно, что подход при интерпретации этих трех типов логических переменных при построении многовходовой модели должен быть разным. Для случая $\alpha(t)$ необходима проверка актуального значения в каждый текущий момент, то есть внутри каждого входа должен полностью при обусловленности некой f_i логическим вектором проверяться весь вектор значений логических переменных.

Для случая $\alpha(t_0)$ необходимо один раз в начале выполнения УА выяснить, к какому варианту следует прибегнуть при исполнении, то есть один раз в момент времени t_0 проверить все логические условия, фигурирующие в логических векторах, обусловливающих выполнение всех f_i, входящих в данный УА.

Для случая $\alpha_{стат}$ проверку можно проводить в любой удобный момент и соответствующим удобным и гибким образом распределять проверки логических условий между входами и организовывать передачи управления между входами в многовходовой модели.

Так, для случаев $\alpha(t_0)$ и $\alpha_{стат}$ можно считать, что от входа к входу логическая обусловленность «наследуется», и образуется цепочка входов, логически обусловленная так же, как отдельная функциональная задача f_i в модели семантики УА. При этом распределенная во времени цепочка входов является аналогом логической ветви исполнения обычного алгоритма, не выполняющегося в реальном времени.

В рамках CASE-системы автоматизированного проектирования УА РВ применяются файлы описания функциональных программ (специальный формат .ops), в которых содержатся данные по временным характеристикам исполнения отдельных действий. Это дает возможность подсчета временных характеристик линейных участков путем суммирования времен исполнения ассемблерных команд передач управления, выдачи команд управления (КУ) бортовой аппаратуре, а также времени выполнения функциональных программ, исполняемых с возвратом. При этом необходимо учесть все возможные варианты выполнения (маршруты) программы управления, определяемые значениями вектора логических переменных $\bar{l} = (\alpha_1, \alpha_2, ..., \alpha_M)$. Необходимо рассматривать логику выполнения алгоритма, и тогда максимальное время выполнения входа определится формулой

$$\tau_{\text{вх max}} = \max_{\substack{\text{по всем} \\ \text{маршрутам}}} \sum_{i=1}^{N} \tau_{\text{лу}\,i}, \qquad (10)$$

где N – число линейных участков на маршруте (варианте) исполнения входа, $\tau_{\text{лу}\,i}$ - длительность i-го линейного участка.

В свою очередь, длительность выполнения линейного участка определяется суммарным временем выполнения присутствующих на нем операций (функциональных программ):

$$\tau_{\text{лу}} = \sum_{j=1}^{L} \tau_j, \qquad (11)$$

где L – число операций линейного участка, τ_j - время исполнения j-ой команды или функциональной программы.

В более крупномасштабном рассмотрении можно подсчитывать также временные характеристики УА в целом, по всем входам. В таком случае для описания передач управления при различных значениях компонент логического вектора можно применить в качестве модели ациклический граф, в котором вершинами являются входы УА, а на-правленные дуги соответствуют передачам управления между входами, как показано на рис. 3.

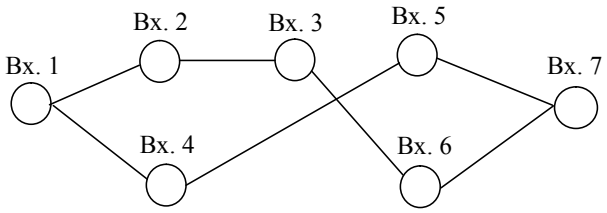

Рис. 3. Логическая структура УА

Выше описан вариант многовходовой модели УА, при котором точка передачи управления из каждого входа фиксирована. Модель становится более сложной при учете возможности передачи управления из произвольного места внутренней структуры входа.

Та же самая модель может быть использована и для представления логической структуры передачи управления внутри входа. Отличия заключаются только в том, что в данном случае на граф передач управления накладывается дополнительное ограничение, а именно: он должен представлять собой дерево с одним корнем.

Таким образом, при применении технологии ГРАФКОНТ возможен автоматический синтез временных характеристик исполнения построенных алгоритмов управления, легкость и оперативность сопоставления вариантов при внесении изменений в управляющие алгоритмы.

Технология ГРАФКОНТ поддерживается специально разработанной программной системой, функционирующей на платформе Windows 95/98/2000/XP и разработанной в основном средствами языка C/C++.

При этом человек-проектировщик, исходя из материалов по логике управления, которую должен реализовывать алгоритм, на входе системы осуществляет интерактивное “конструирование” визуального образа в соответствующей графической подсистеме.

Выходными данными системы является техническая документация на УА РВ, требуемая согласно принятой на предприятии-заказчике системе, а именно: временная диаграмма управляющего алгоритма и блок-схема программы, а также собственно текст управляющей программы на языке автокода бортовой ЦВМ, реализующей УА РВ.

Говоря о методах обеспечения надежности программ путем их тестирования, необходимо отметить, что несмотря на повышение надежности проектируемого алгоритма при использовании технологии и программного комплекса ГРАФКОНТ, система не может парировать грубые семантические ошибки, допущенные при конструировании УА в графическом конструкторе, например, пробелы или упущение раздела в материалах по логике управления. Поэтому нецелесообразно отказываться от комплексного поэтапного тестирования полученного программного продукта, в ходе которого путем моделирования бортового функционирования УА выявляется его соответствие исходным требованиям.

На протяжении длительного периода времени на предприятии-заказчике (ГНПРКЦ “ЦСКБ-Прогресс”, г. Самара) выработана методика поэтапной отладки управляющих программ на специальном наземном отладочном моделирующем комплексе, в который входят ЭВМ, эмулирующие БВС и ЭВМ для моделирования внешних воздействий космического пространства и других факторов среды [4].

Отладка включает автономную отладку и комплексную отладку. При этом для каждой управляющей программы составляется отладочное задание на специальном символьном языке отладки. В отладочное задание входят перечень отслеживаемых по значению переменных, необходимых точек останова и т. д.

Созданная система автоматизированной генерации отладочных заданий ГЕОЗ позволяет на базе внутренних структур данных программного комплекса ГРАФКОНТ

автоматически формировать отладочное задание на автономную отладку УА.

Важными задачами при проведении отладки УА РВ являются автоматизированное выявление всех возможных вариантов исполнения («маршрутов») алгоритма в зависимости от значений компонент вектора логических переменных и выявление всех имеющихся в алгоритме информационных и управляющих связей с другими управляющими программами. Эти задачи также решаются в рамках программной системы ГЕОЗ.

Список литературы

1. “Авиастроение”. Том 6 (Итоги науки и техники, ВИНИТИ АН СССР). М., 1978.

2. Логика и компьютер. Моделирование рассуждений и проверка правильности программ / А. М. Анисов, П. И. Быстров, В. А. Смирнов и др. М.: Наука, 1990.

3. Тюгашев А. А. Проблема неоднозначности при порождении логико-временной структуры управляющего алгоритма по многовходовой модели реального времени. // Сб. трудов Третьей международной молодежной школы-семинара БИКАМП-01, СПб, 2001.

4. Управление космическими аппаратами зондирования Земли: Компьютерные технологии / Д. И. Козлов, Г. П. Аншаков, Я. А. Мостовой, А. В. Соллогуб. М.: Машиностроение, 1998.

5. Калентьев А. А., Тюгашев А. А. Разработка подсистемы синтеза управляющих алгоритмов на базе исчисления УА // Всероссийская научная шклола “Компьютерная алгебра, логика и интеллектное управление. Проблемы анализа стратегической стабильности”: Сб. трудов, Иркутск, ИрВЦ СО РАН, 1994.

6. Калентьев А.А. Автоматизированный синтез алгоритмов асинхронного управления технологическими системами с множеством дискретных состояний. Самара: Самар.аэрокосм.ун-т., 1998.

METHOD OF DESIGNING RELIABLE REAL-TIME CONTROL ALGORITHMS FOR SPACECRAFT

© 2004 A. A. Tyugashev

Samara State Aerospace University

The problem of designing reliable real-time control algorithms mounted aboard spacecraft is under consideration. The method of designing control algorithms based on the algorithm semantics model is described which guarantees higher reliability. Data on a computer-aided design system that support the technology mentioned are given.