стереотипных схем в диссонанс может быть формализована в виде известной задачи о рюкзаке:

$$\sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij}x_j \to \max,\ \sum_{j=1}^{m} r_j x_j \le R,$$

или, альтернативно, в виде

$$\sum_{j=1}^{m} r_j x_j \to \min,\ \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij}x_j \ge C,$$

где R – имеющийся запас ресурсов на реализацию системы; C – минимальный требуемый уровень эффективности; x_i – переменная принятия решения, равная единице, если i-й способ приведения стереотипных схем в диссонанс включается в формируемый набор, и равная нулю в противном случае. Решение данной задачи является чисто технической проблемой.

В более сложных случаях существует не один ресурс, затрачиваемый на реализацию защиты, и эффективность определяется не одним числом (или вообще не числом). Более того, возможны конфликты между способами защиты, когда использование одного способа исключает или изменяет эффективность применения другого. Возможно также, что эффективность и затраты ресурсов одного и того же способа защиты будут разными для различных способов атаки. Все это может значительно усложнить постановку задачи выбора эффективного варианта, приводя к нелинейным многокритериальным задачам оптимизации с разношкальными переменными и алгоритмически заданными функциями. Решение такой задачи уже не является чисто технической проблемой, но, тем не менее, вполне может быть осуществлено с помощью современного алгоритмического аппарата оптимизации [2].

Библиографический список

1. Эшби, У. Р. Введение в кибернетику / У. Р. Эшби. М. : КомКнига, 2006.

2. Семенкин, Е. С. Метод обобщенного адаптивного поиска для синтеза систем управления сложными объектами / Е. С. Семенкин, В. А. Лебедев. М. : МАКС Пресс, 2002.

E. S. Semenkin, M. A. Styugin

PROTECTION AGAINST INVESTIGATION AND APPLICATION IN SECURITY SYSTEMS

Informational interaction of a violator and a protected system in the form of stereotypic investigation schemes is considered. The attack effectiveness decreasing method based on artificial increase of the system diversification is suggested. Security systems design automation problem is described.

Keywords: informational interaction with a violator, stereotypic schemes, risk reducing in security systems.

УДК 004.056

И. А. Капчинский, П. В. Ковалев, А. Н. Лайков, С. Н. Гриценко

К ВОПРОСУ ФОРМИРОВАНИЯ МУЛЬТИВЕРСИОННОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ С УЧЕТОМ РЕСУРСНЫХ ОГРАНИЧЕНИЙ

Рассматривается методология мультиверсионного программирования, которая обеспечивает гарантию того, что ошибки одной из версий программного обеспечения не приведут к нарушению процесса управления сложными объектами, для которых характерны жесткие требования по надежности и автономности функционирования.

Ключевые слова: оптимизация, надежность, мультиверсионное программирование.

Проблеме формирования программных комплексов (ПК), проектируемых на основе принципов программной избыточности, в настоящее время уделяется значительное внимание. Проблематика проектирования программных комплексов с использованием методологии мультиверсионного программирования рассматривалась в работах А. Авижиениса, Н. Ашрафи, О. Бермана, М. Катлер, Дж. Ву, К. Яо, Р.К. Скотта, Д. Мак Аллистера, К. Е. Гросспитча и мн. др. [1–5]. Разрабатываются новые методы оптимизации версионного состава программного комплекса, новые системы формирования структуры программного комплекса, однако не достаточно внимания уделяется разработке методик формирования структуры мультиверсионного программного комплекса с учетом временных и ресурсных ограничений [1]. Основной задачей формирования мультиверсионного программного обеспечения является оптимизация версионного состава программного комплекса. В качестве критерия оптимизации обычно выбирают надежность комплекса.

В научных исследованиях, например, в работе [2] уже рассматривалось введение ограничений на ресурсы и время выполнения программы, но предложенный алгоритм требует больших вычислений и сложен в реализации. Кроме того, если временные ограничения вводятся

в полном объеме, то для ресурсных ограничений рассматривается частный случай, который не отражает существующую реальность. Фактически, возможно применение только одной единственной оценки надежности программного комплекса и использование одноуровневой модели построения ПО.

Методология авторов работ [3; 4], исследующих данный вопрос, основывается на следующих положениях:

– ПО проектируется методом модульного программирования;

– рассматриваются системы, построенные на основании методологии мультиверсионного программирования;

– версии модулей разрабатываются независимо друг от друга и их параметры надежности и стоимости могут быть оценены;

– реализуется статическое распределение задач с известным временем обслуживания.

В работе рассматривается следующая модель: система конструируется из I задач, для решения каждой из которых назначается определенный программный модуль. Связи между задачами определяются в виде графа задач и также могут быть представлены в виде матрицы инцидентности. Некоторые из задач могут быть структурно идентичны или одинаковы, что позволяет для их решения использовать одинаковые модули (цикличное использование модулей или эксплуатация одного и того же модуля с разными параметрами). Пусть J – число используемых программных модулей. Следовательно, I задач можно разбить на J классов. Через $n_j, j = 1, \ldots, J$ обозначим число задач класса j, а через A_j – множество индексов задач этого класса

$$card\, A_j = n_j, j = 1, \ldots, J. \quad (1)$$

Предполагается, что, учитывая методологию мультиверсионного программирования, имеется K_j версий модуля j, используемого для решения j-го класса задач ($j = 1,\ldots, J$), т. е. степень избыточности может варьироваться в зависимости от критичности задачи или имеющегося набора версий. Для формального описания использования нескольких версий одного программного модуля вводится следующая булева переменная [5; 6]: $X_{ij}^{kj} = 1$, если для решения задачи i класса j используется k_j-я ($k_j = 1, \ldots, Kj$) версия модуля j, и 0 – в противном случае.

Множество $X = \left(X_{ij}^{kj}\right), i = 1, \ldots, I; j = 1, \ldots, J; k_j = 1, \ldots, K_j$ булевых переменных X_{ij}^{kj} называется вектором конфигурации системы и представляет возможный способ назначения версий программных модулей задачам в системе.

Вводятся следующие обозначения:

– R_{kj} – оцениваемая надежность версии k_j модуля j: ($R_{kj} = 1 - P_{kj}$), где P_{kj} – соответствующая функция распределения отказов;

– R_j – оцениваемая надежность отказоустойчивого модуля j как комбинации K_j независимых версий;

– R – оцениваемая надежность всего комплекса ПО.

Условие реализации задачи класса j одной из версий модуля в терминах булевых переменных может быть выражено следующим образом:

$$\sum_{i_j \in A_j} X_{ij}^{kj} = n_j, j = 1, \ldots, J. \quad (2)$$

Тогда задача состоит в максимизации надежности

$$\max R = \prod_{j=1}^{J} R_j, \quad (3)$$

где

$$R_j = 1 - \prod_{k_j=1}^{K_j} \left(1 - R_{kj}\right)^{X_{ij}^{kj}}$$

с дополнительными ограничениями

$$\sum_{i_j \in A_j} X_{ij}^{kj} \geq n_j, j = 1, \ldots, J. \quad (4)$$

Здесь выражение (4) определяет то, что в мультиверсионной системе для некоторой задачи может использоваться более одной версии программного модуля.

С целью описания процессов обслуживания задач в системе вводится так называемый вектор временной развертки (ВВР) набора задач, определяемый как

$$t = \{t_1, \ldots, t_i, \ldots, t_I\},$$

где t_i – момент времени, в который начинается обслуживание i-й ($i = 1, \ldots, I$) задачи. Независимо друг от друга компоненты ВВР должны удовлетворять условиям

$$t_i^l \leq t_i \leq t_i^u \forall i = 1, \ldots, I,$$

где t_i^l и t_i^u – самый ранний и самый поздний моменты времени соответственно, когда может начаться обслуживание задачи i; оба параметра зависят от значений времени выполнения предшествующих задач. Очевидно, что временные критерии улучшаются варьированием параметра t_i в рамках величин t_i^l и t_i^u.

Блок-схема алгоритма формирования ВВР ресурсов изображена на рис. 1. Алгоритм заключается в последовательном переборе всех путей следования в порядке убывания времени их выполнения и обработки каждого модуля в цепочке. Каждый модуль, во-первых, проверяется на его включенность в ВВР, далее определяется временной промежуток, в течение которого предполагается выполнение модуля.

Алгоритм поиска начального и конечного момента времени представлен в виде блок-схемы на рис. 2. В качестве начального момента времени используем момент времени, когда заканчивается выгрузка параметров последних из модулей, от которых зависит текущий. Это соответствует моменту времени, когда мы можем загружать параметры, поэтому мы корректируем начальное время, уменьшая его на время загрузки модуля, которое нам известно заранее [7].

Здесь $\tau^*_i, i = 1 \ldots n_r^m$ – параметры вектора временной развертки; k – версия программного модуля; N_m – программные звенья; n_λ – длина пути; λ_i – номер выполняемого звена; $(a^*)^1_{kj}$ и $(a^*)^1_{ik}$ – элементы матрицы связанности A, структура которой изображена на рис. 3.

Операция считается удачной, если нашлось такое t_{start}, для которого для всех необходимых ресурсов имеется достаточное количество свободного объема до нарушения временного ограничения. В случае, когда время окончания загрузки превышает t_{end}, необходимо исключить из ВВР ресурсов все зависимые от текущего модуля отрезки цепочек, учтенные ранее.

Программный комплекс имеет N_{in} входов и N_{out} выходов, в матрице A в качестве элементов a_{ij} будем использовать структуру $\{in_j, out_i\}$, где in_j – номер входного пара-

метра j-го звена, а out_i – номер выходного параметра i-го звена, если существует передача данных, и {0, 0}, если нет. Полученную матрицу обозначим A_t^*, а обращаться к параметрам in_j и out_i будем при помощи символов $(a_t^*)_{ij}^1$ и $(a_t^*)_{ij}^2$ соответственно. Иначе говоря, мы представляем выходные параметры ПК как задачи, которые могут получить один параметр, но ничего не передают, а входные параметры ПК наоборот передают, но не получают.

Предлагается реализовать следующую процедуру оптимизации расписания задач в мультиверсионной системе путем пошаговой коррекции компонент ВВР. Для этого следует ввести еще один вектор, состоящий из I компонент – вектор реализации (ВР). Компонента h_{ij}^{kj} вектора реализации представляет собой время, необходимое на выполнение задачи i средствами k_j-й версии программного модуля j.

Программные модули назначаются в соответствии со следующими правилами [7]:

1. Реализуемость (выполнимость) прикладного процесса.

2. Коррекция компонент ВВР в ответ на нарушение условия реализуемости.

Кроме учета ограничений стоимостного и временного характера при проектировании оптимального состава ПО следует принимать во внимание ограничения на ресурсы. В общем случае используется предположение о том, что задача использует ресурсы всех необходимых видов на протяжении всего ее выполнения. В работе рассматриваются два вида ресурсов – активные и пассивные. Ресурсы называют активными, если они обладают вычислительными возможностями (процессор, устройства ввода/вывода), в противном случае ресурсы называют пассивными (файлы, архивы, базы данных). Для выполнения любой задачи требуется не менее одного активного ресурса и любое число (как и их полное отсутствие) пассивных ресурсов. Таким образом, пассивные ресурсы должны использоваться вместе с активными. Некоторые виды ресурсов могут использоваться одновременно несколькими задачами, в то время как другие (например, каналы считывания/записи) можно назначать только одной задаче.

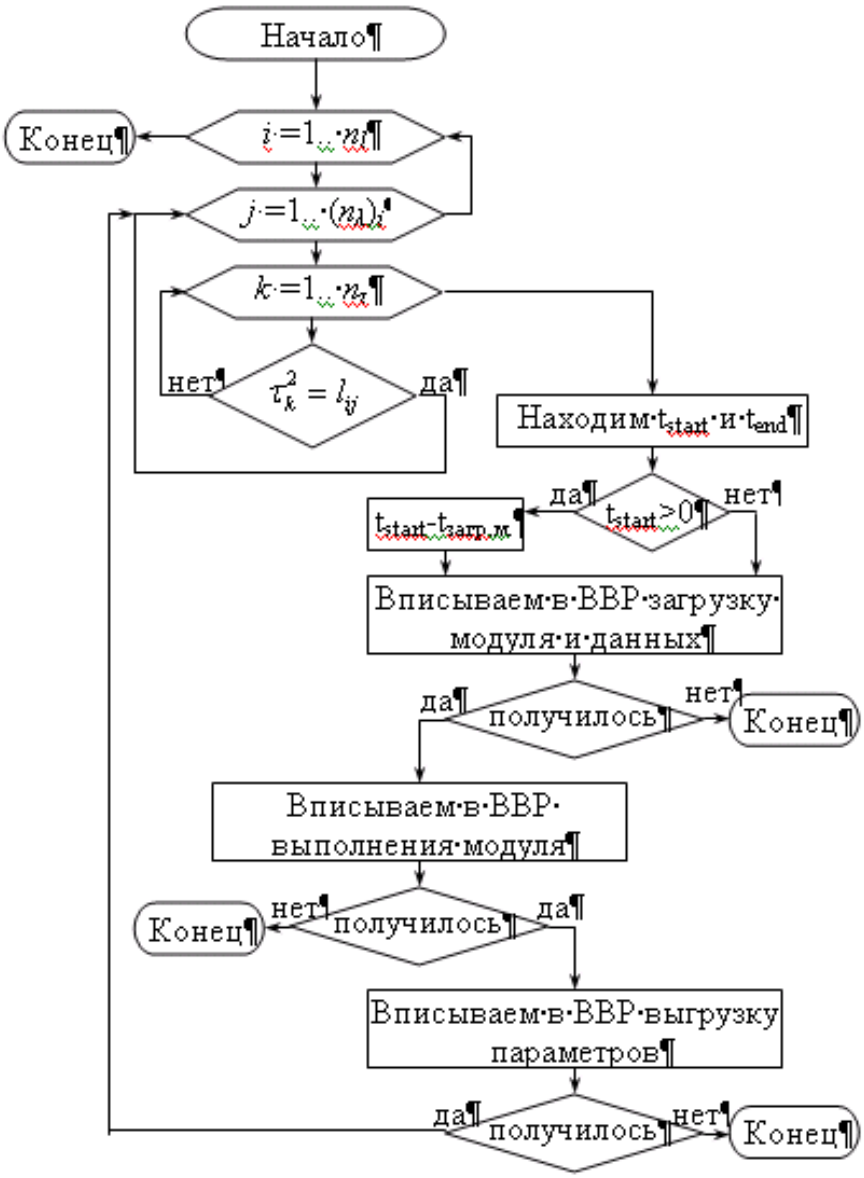

Рис. 1. Обобщенная блок-схема алгоритма формирования ВВР-ресурсов

Основываясь на введенных понятиях, строится один из алгоритмов назначения модулей.

1. Назначаем k_j-й программный модуль для решения задачи $i = 1$ класса j.

2. Проверка процесса на реализуемость:

– если условие выполняется – переход к п. 4;

– если условие не выполняется – переход к п. 3.

3. Коррекция вектора временной развертки:

– если условия выполняются – продолжаем далее по ходу алгоритма;

– если условия не выполняются – переходим к п. 1.

4. Проверка ограничений на число каналов чтения/записи:

– если условия выполняются – переход к п. 5;

– если условия не выполняются – переход к п. 1.

5. Увеличиваем i: ($i = i + 1$) и назначаем k_j-й программный модуль для решения задачи i класса j:

– если $i + 1 \le I$ – переход к п. 2;

– если $i + 1 > I$ – переход к п. 6.

6. Вычисление длительности вычислительного процесса для измененного способа назначения модулей.

Рассматривая в рамках данной процедуры методы оптимизации версионного состава по критерию надежности ПК [7], следует отметить, что описанный выше алгоритм с точки зрения программиста является нереализуемым. Во-первых, символом k_j в работе обозначается номер версии в наборе для модуля j, а не номер модуля, а во-вторых, в алгоритме изменяется только параметр i, а параметр k_j остается неизвестным.

В представленном алгоритме также отсутствует проверка на занятость активных и пассивных ресурсов, не являющихся каналами чтения и записи. Заранее знать временной интервал выполнения каждой задачи не всегда возможно, тем более, если учитывать то, что от на-

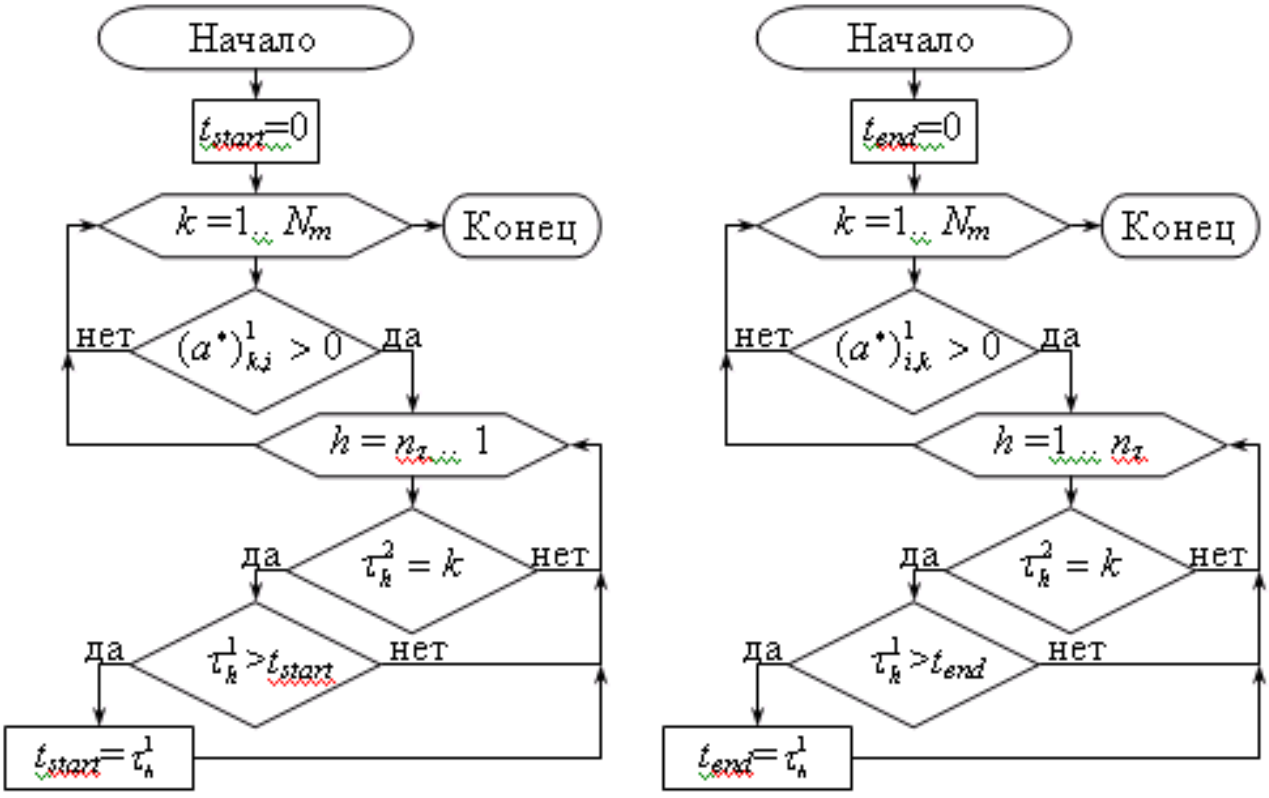

Рис. 2. Блок-схемы алгоритма нахождения t_{start} и t_{end}

		Задачи				Выходные параметры ПК			
		1	2	…	N_t	N_t+1	N_t+2	…	N_t+N_{out}
Задачи	1	$\{in_1, out_1\}$	$\{in_2, out_1\}$	…	$\{in_{Nt}, out_1\}$	$\{1,out_1\}/\{0,0\}$	$\{1,out_1\}/\{0,0\}$	…	$\{1,out_1\}/\{0,0\}$
	2	$\{in_1, out_2\}$	$\{in_2, out_2\}$	…	$\{in_{Nt}, out_2\}$	$\{1,out_2\}/\{0,0\}$	…	…	…
	…	……							
	N_t	$\{in_1, out_{Nt}\}$	$\{in_2, out_{Nt}\}$	…	$\{in_{Nt}, out_{Nt}\}$	$\{1,out_{Nt}\}/\{0,0\}$	$\{1,out_1\}/\{0,0\}$	…	$\{1,out_{Nt}\}/\{0,0\}$
Входные параметры ПК	N_t+1	$\{in_1,1\}/\{0,0\}$	$\{in_2,1\}/\{0,0\}$	…	$\{in_{Nt},1\}/\{0,0\}$	{0,0}	{0,0}	…	{0,0}
	N_t+2	$\{in_1,1\}/\{0,0\}$	…	…	…	{0,0}	{0,0}	…	{0,0}
	…	……							
	N_t+N_{in}	$\{in_1,1\}/\{0,0\}$	…	…	$\{in_{Nt},1\}/\{0,0\}$	{0,0}	{0,0}	…	{0,0}

Рис. 3. Структура матрицы связности задач ПК

бора версий напрямую зависит время выполнения задачи.

Таким образом, становится ясным, что основная цель дальнейших исследований – показать возможность и эффективность использования алгоритма формирования ВВР ресурсов при синтезе оптимального состава мультиверсионного ПО.

В заключение следует отметить, что использование ресурсных и временных ограничений при формировании мультиверсионного ПК требует дальнейшего более детального рассмотрения. Рациональное структурное построение программных комплексов гарантирует достаточно полное использование ресурсов ЭВМ. Однако, технологические особенности проектирования ПК, дополняя проблему структурного проектирования ПО, выводят ее в разряд общих проблем разработки методов и автоматизированных систем проектирования сложных программных комплексов.

Библиографический список

1. Майерс, Г. Надежность программного обеспечения / Г. Майерс ; пер. с англ. Ю. Ю. Галимова / под ред. В. Ш. Кауфмана. М. : Мир, 1980.

2. Kovalev, I. System of Multi-Version Development of Spacecrafts Control Software / I. Kovalev. Pro Universitate Verlag Sinzheim, 2001.

3. Avizienis, A. On the Implementation of N-Version Programming for Software Fault Tolerance During Execution / A. Avizienis, L. Chen // Proceedings of the IEEE COMPSAC'77. 1977. P. 149–155.

4. Randell, B. The Evolution of the Recovery Block Concept / B. Randell, J. Xu // Software Fault Tolerance ; ed. by M. R. Lyu. Wiley, 1995.

5. Kovalev, I. Deriving the optimal Structure of N-Version Software under Resource Requirements and Cost/Timing Constraints / I. Kovalev, K.-E. Grosspietsch // Proceedings of Euromicro'2000. Maastricht, 2000. P. 200–207.

6. Kovalev, I. Optimization models for reliability of telecommunication software systems / I. Kovalev, A. Popov, Ju. Shipovalov // Advances in Modeling & Analysis. Series B. 2000. Vol. 43, № 3–4. P. 41–46.

7. Ковалев, И. В. Мультиверсионный метод повышения программной надежности информационно-телекоммуникационных технологий в корпоративных структурах / И. В. Ковалев, Р. В. Юнусов // Телекоммуникации и информатизация образования. 2003. № 2 (15). С. 50–55.

I. A. Kapchinsky, P. V. Kovalev, A. N. Laykov, S. N. Gritsenko

ON MULTIVERSION SOFTWARE FORMATION CONSIDERING THE RESOURCE CONSTRAINTS

The multiversion programming methodology which guarantees that mistakes of one software versions will not lead to infringement of the control process by complex objects where rigid requirements for reliability and autonomy of functioning are typical.

Keywords: optimization, reliability, multiversion programming.

УДК 621.393.3

В. Б. Малинкин, Е. В. Кулясов, Е. В. Малинкин, И. И. Павлов

АСИММЕТРИЧНЫЙ ИНВАРИАНТНЫЙ ЭХО-КОМПЕНСАТОР ВТОРОГО ПОРЯДКА БЕЗ ЗАЩИТНОГО ВРЕМЕННОГО ИНТЕРВАЛА И ЕГО ХАРАКТЕРИСТИКИ

Синтезирована структура инвариантного эхо-компенсатора второго порядка без защитного временного интервала. Приведены элементы управления подобным эхо-компенсатором и расчет его основных технических характеристик.

Ключевые слова: инвариантный, эхо-компенсатор, асимметричный, мощность собственных шумов.

Магистральные телекоммуникационные сети в подавляющем большинстве случаев используют волоконно-оптические системы передачи. Однако сдерживающим фактором доведения стандартных потоков до потребителя является так называемая «последняя миля», представленная медным кабелем.

Существующие высокоскоростные технологии доступа используют в оборудовании xDSL-адаптивные эхо-компенсаторы. Принцип их работы основан на моделировании параметров неизвестной системы, а основным недостатком является большая критичность работы к корреляционным связям сигналов двух направлений. В некоторых случаях это может привести даже к срыву работы.

Другой подход к построению адаптивных эхо-компенсаторов состоит в использовании инвариантов. Так, в работах [1; 2] приведены структуры инвариантных эхо-компенсаторов с защитным и временным интервалом и без него. Основным отличием инвариантных эхо-компенса-