УДК 004.42

МОДЕЛИ ГЕРТ-СЕТЕЙ ДЛЯ РАЗЛИЧНЫХ СПОСОБОВ ПРИМЕНЕНИЯ МЕТОДОЛОГИИ МУЛЬТИВЕРСИЙ*

И. В. Ковалев, П. В. Зеленков, М. В. Сарамуд., Г. А. Сидорова, В. В. Брезицкая

Сибирский государственный аэрокосмический университет имени академика М. Ф. Решетнева
Россия, 660014, г. Красноярск, просп. им. газ. «Красноярский рабочий», 31. E-mail: kleniks@yandex.ru

Описываются различные способы применения методологии мультиверсий при разработке отказоустойчивого программного обеспечения, а также представлены модели мультиверсионных программных архитектур в виде базовых ГЕРТ-сетей. С учетом необходимого и достаточного условия функционирования мультиверсионного модуля выбран узел с IOR входом и детерминированным выходом для представления участка сети, описывающего данный модуль. Приводятся формулы расчета вероятностно-временных характеристик мультиверсионных модулей, обеспечивающие пригодность ГЕРТ-сетевых моделей для описания мультиверсионного программного обеспечения. Это позволяет использовать эквивалентные преобразования и существующие методы расчета ГЕРТ-сетей.

Ключевые слова: стохастическая сеть, вероятностно-временной анализ, мультиверсионность, отказоустойчивость, программное обеспечение.

MODEL OF GERT-NETWORKS FOR DIFFERENT WAYS OF USAGE OF MULTI-VERSION METHODOLOGY

I. V. Kovalev, P. V. Zelenkov, M. V. Saramud, G. A. Sidorova, V. V. Brezitskaya

Siberian State Aerospace University named after academician M. F. Reshetnev
31 “Krasnoyarskiy Rabochiy” prospect, Krasnoyarsk, 660014, Russia. E-mail: kleniks@yandex.ru

This article describes different ways of application of multi-version methodology in the development of multi-version fail-safe software, and presents models of multi-version software architectures in the form of basic GERT- networks. With respect to the necessary and sufficient condition for the operation of multi-version module, the node with IOR input and output is selected to provide deterministic network section that describes the module. Formulas for calculation of probabilistic-time characteristics of multi-version modules, that provide fitness of GERT-network models to describe multi-version software, are presented. This allows to use the equivalent conversion and existing methods of calculation of GERT- networks.

Keywords: stochastic network, probabilistic-timing analysis, multi-versioning, fail safety, software.

Мультиверсионная отказоустойчивость основана на использовании двух или более версий (или вариантов) модуля программного обеспечения (ПО), исполняемых последовательно или параллельно. Версии используются как альтернативы (с отдельными средствами обнаружения ошибок), в парах (чтобы осуществить обнаружение проверками дублирования) или в больших группах (чтобы маскировать ошибки через голосование) [1]. Использование множественных версий обосновывается предположением о том, что поразному построенные компоненты (т. е. различными проектировщиками, различными алгоритмами, различными инструментальными средствами проектирования и т. д.) должны иметь разные ошибки. Поэтому если одна версия производит сбой на специфическом вводе, то по крайней мере одна из альтернативных версий должна обеспечить корректный вывод [2]. В данной статье описываются различные способы применения методологии мультиверсий для разработки отказоустойчивого ПО, а также приводятся модели мультиверсионных программных архитектур в виде ГЕРТ-сетей.

1. **Необходимое и достаточное условие функционирования мультиверсионного модуля отказоустойчивого ПО.** Необходимым и достаточным условием функционирования мультиверсионного модуля является выполнение хотя бы одной мультиверсии [3]. Поэтому в общем случае участок сети описывающий мультиверсионный блок программного обеспечения, будет иметь детерминированный выход и узел с IOR-входом. Выбор данного типа узла связан с тем, что этот узел по определению активируется при выполнении любой дуги, входящей в него. Строго говоря, данный узел с IOR входом не будет IOR узлом в классическом понимании, а служит лишь для отображения случая выполнения необходимого и достаточного условия функционирования мультиверсионного модуля (выполнения единственной мультиверсии)

* В рамках государственного задания высшим учебным заведениям 8.5534.2011.

и в конечном счете вся подсеть, используя эквивалентное преобразование, будет заменена одним блоком.

Рассмотрим расчет сети, описывающей мультиверсионную архитектуру ПО, которая содержит IOR вход и детерминированный выход (рис. 1) и состоит из нескольких подсетей (рис. 2).

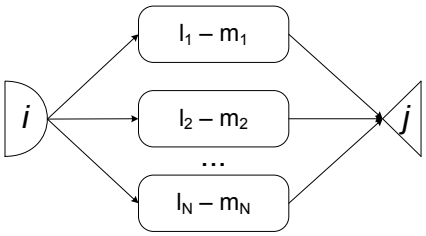

Рис. 1. IOR-вход

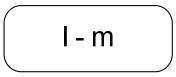

Рис. 2. Произвольная подсеть с начальным узлом *l* и конечным узлом *m*

Необходимо учитывать, что j – вычисляемый IOR-вход, имеющий стохастическое начало в узле i. $\{l_1 - m_1\}$, $\{l_2 - m_2\}$, …, $\{l_N - m_N\}$ – N непересекающихся подсетей [4].

При увеличении числа модулей в мультиверсионной подсети вероятность отказа всех модулей будет снижаться, а вероятность того, что хотя бы один из модулей сработает, будет расти. Также следует учесть, что время работы N модулей будет не больше времени работы $t_{\max}$, где $t_{\max}$ – максимальное время работы мультиверсионного модуля, входящего в набор N.

Пусть p_{ij} – вероятность того, что операция (i, j) будет выполнена при условии, что узел i выполнен и $F_{ij}(t)$ – функция условной вероятности распределения времени выполнения работы (i, j), тогда имеем

$$p_j = p_i[1-(1-p_{l1,m1\kappa})(1-p_{l2,m2\kappa})\cdot\ldots\cdot(1-p_{l_N,m_N\kappa})\,], \quad (1)$$

$$F_j(t) = \max(t_{l1,m1\kappa}, t_{l2,m2\kappa}, ..., t_{l_N,m_N\kappa}). \quad (2)$$

Следует также учитывать, что при увеличении числа мультиверсионных модулей в системе, будет возрастать время на межмодульное взаимодействие. Другими словами, при организации параллельной работы N модулей время, затраченное системой на контроль и распараллеливания процессов, будет возрастать и, в зависимости от конкретной реализации среды исполнения [5], может оказывать существенное влияние на работу системы в целом. Не углубляясь в методику расчета и теоретическое обоснование этой величины, учтем ее влияние на систему следующим образом:

$$t'_{\max} = t_{\max} + \Delta t, \quad (3)$$

где $t_{\max}$ – время работы мультиверсионного участка ГЕРТ-сети; Δt – время, затраченное системой на межинтерфейсное взаимодействие.

Введем следующие обозначения:

$M(X)$ – математическое ожидание времени работы мультиверсионного блока, модули которого выполняются параллельно;

x_i – время работы i-го модуля (математическое ожидание времени работы мультиверсионного модуля i);

n – количество мультиверсионных модулей в блоке.

Тогда математическим ожиданием времени работы мультиверсионного блока предлагается считать максимальное математическое ожидание среди всех n мультиверсионных модулей:

$$M(X) = \max_{1\le i\le n}(x_i)\,. \quad (4)$$

Введем дополнительные обозначения:

$D(X)$ – дисперсия времени работы мультиверсионного блока, модули которого выполняются параллельно;

$\sigma(X)$ – стандартное отклонение или среднеквадратическое отклонение времени работы мультиверсионного блока ($\sigma(X) = \sqrt{D(X)}$).

Стандартное отклонение времени работы мультиверсионного блока предлагается определять следующим образом.

Шаг 1. Определить максимальную величину $M(x_i) + \sigma(x_i)$ среди всех n блоков (максимально возможное отклонение).

Шаг 2. Определить максимальное математическое ожидание среди всех n мультиверсионных модулей.

Шаг 3. Разность п. 1 и п. 2 и будет стандартным отклонением времени работы мультиверсионного блока.

$$\sigma(X) = \max_{1\le i\le n}(M(x_i) + \sigma(x_i)) - \max_{1\le i\le n}(x_i), \quad (5)$$

$$D(X) = \sigma(X)^2. \quad (6)$$

2. Применение методологии мультиверсий в виде базовых ГЕРТ-сетевых моделей

***N*-версионное программирование.** *N*-версионное программирование – это мультиверсионная методика, в которой все версии разработаны в соответствии с идентичными основными требованиями, а решение о правильности вывода основано на сравнении всех выводов [6] (рис. 3, 4).

Использование универсального алгоритма выбора решения (обычно голосования) для выбора правильного вывода – фундаментальное различие этого подхода и подхода с блоком восстановления, который требует зависимого от приложения приемочного теста. Так как все версии построены в соответствии с идентичными требованиями, то использование *N*-версионного программирования требует значительно больших затрат при разработке, но сложность, т. е. трудность разработки не обязательно будет намного большей, чем сложность формирования одной версии.

Проектирование выбирающего алгоритма может быть усложнено необходимостью реализовать алгоритм неточного голосования [1]. Многие исследования направлены на разработку методологий, которые увеличи-

вают вероятность достижения эффективного разнообразия в конечном продукте. Фактическое выполнение версий может быть последовательным или параллельным. Последовательное выполнение может потребовать использования контрольных точек, чтобы перезагрузить состояние перед выполнением дополнительной версии.

Блоки восстановления. Методика блоков восстановления комбинирует основные идеи метода контрольных точек и рестарта для мультиверсионных компонентов программного обеспечения таким образом, что различные версии используются только после того, как обнаруживается ошибка [7] (рис. 5, 6).

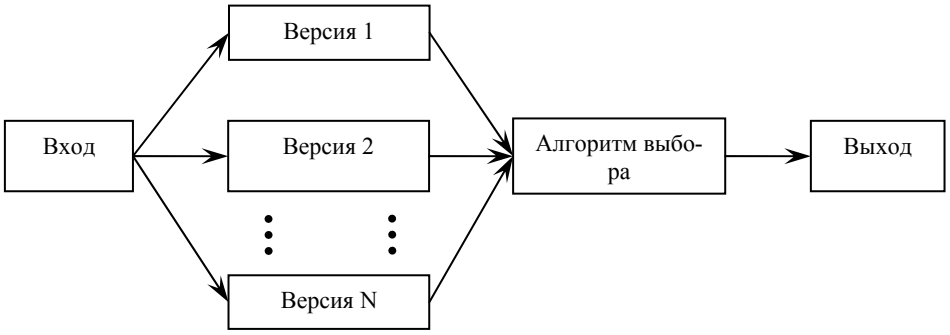

Рис. 3. Модель *N*-версионного программирования

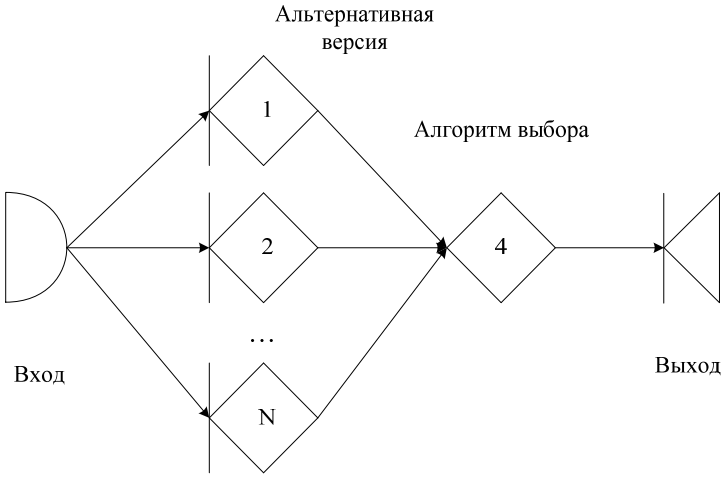

Рис. 4. Модель *N*-версионного программирования в виде ГЕРТ-сети

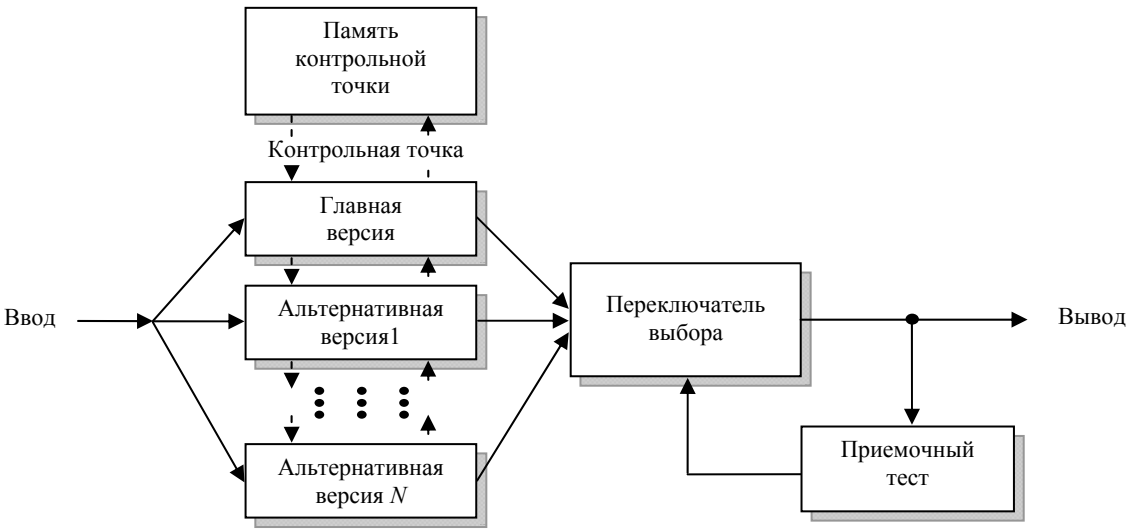

Рис. 5. Модель блока восстановления

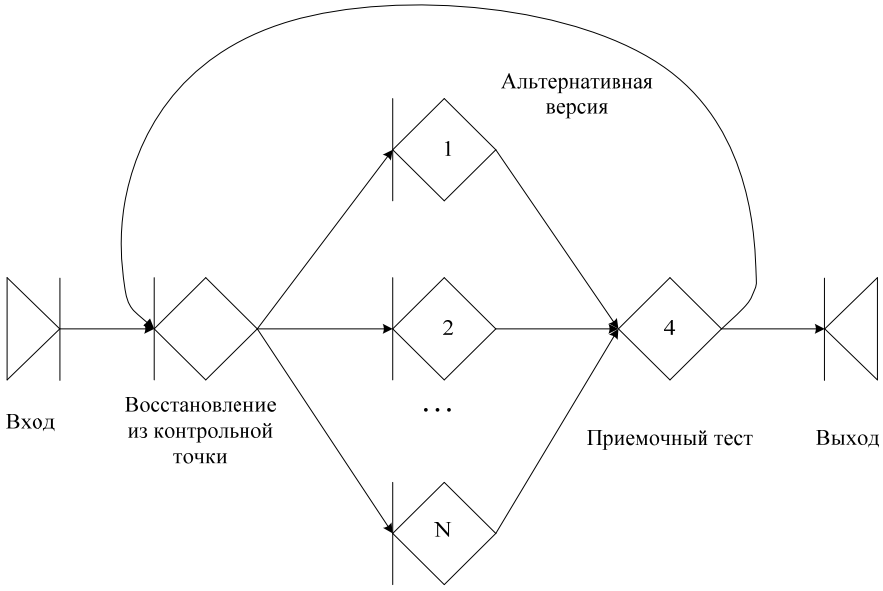

Рис. 6. Модель блока восстановления в виде ГЕРТ-сети

Контрольные точки определяются перед исполнением версий. Контрольные точки необходимы, чтобы восстановить состояние после того, как версия произведет сбой и не сможет обеспечить корректную начальную точку для следующего компонента. Приемочный тест не должен быть только выходным тестом и может осуществляться различными встроенными проверками, чтобы увеличить эффективность обнаружения ошибок. Также, из-за того, что первичная версия будет успешно выполняться в большинстве случаев, альтернативы могут быть разработаны с более низкой производительностью и качеством (в некотором смысле) (например, вычисляя значения с меньшей точностью). Подобно методу разнообразия данных, вывод альтернативных версий может быть разработан таким образом, чтобы быть эквивалентным первичному (с определением эквивалентности, зависящей от приложения).

Фактическое выполнение множественных версий может быть последовательным или параллельным в зависимости от вычислительных мощностей и требований производительности. Если все альтернативы выдали сбой, компонент должен инициировать исключение, чтобы сообщить остальной части системы об отказе завершения его функции.

Следует отметить, что такое возникновение отказа не подразумевает постоянного отказа компонента и он может быть повторно использован после изменений его состояния или ввода. Возможность совпадения отказов является источником больших дискуссий относительно всей технологии отказоустойчивого мультиверсионного программного обеспечения [6].

***N*-версионное программирование с самоконтролем.** *N*-версионное программирование с самоконтролем – это мультиверсионное программное обеспечение, комбинирующее методологию блока восстановления и *N*-версионное программирование. *N*-версионное программирование с самоконтролем использует приемочные тесты (рис. 7, 8). В данном случае версии и приемочные тесты разработаны независимо от общих требований. Здесь используются отдельные приемочные тесты на каждую версию, что является основным различие этой модели от методологии с блоком восстановления. Подобно методологии с блоком восстановления, выполнение версий и их испытание может быть сделано последовательно или параллельно, но вывод принят от версии, которая выбрана голосованием и которая прошла приемочный тест. Последовательное выполнение требует использования контрольных точек, а параллельное выполнение требует использования алгоритмов синхронизации ввода и состояний.

N-версионное программирование с самоконтролем, использующее сравнение с самоконтролем, использующее сравнение для обнаружения ошибок (рис. 9). Подобно *N*-версионному программированию, эта модель имеет преимущество использования алгоритма выбора решения при выборе правильного вывода, который зависит от приложения (рис. 9, 10). Этот вариант программирования имеет теоретическую уязвимость при столкновении с ситуациями, где множественные пары передают различные данные для сравнения. Этот случай должен быть рассмотрен, и соответствующий решающий подход должен быть выбран на стадии проектирования [1].

Блоки восстановления с согласованием. Блоки восстановления с согласованием (рис. 11, 12) – это подход, комбинирующий *N*-версионное программирование и метод блоков восстановления, для повышения надежности по сравнению с использованием только одного из подходов. Согласно [6], приемочные тесты в блоках восстановления страдают из-за недостатка руководящих принципов для их разработки и общей склонности к ошибкам проектирования из-за трудности создания эффективных тестов.

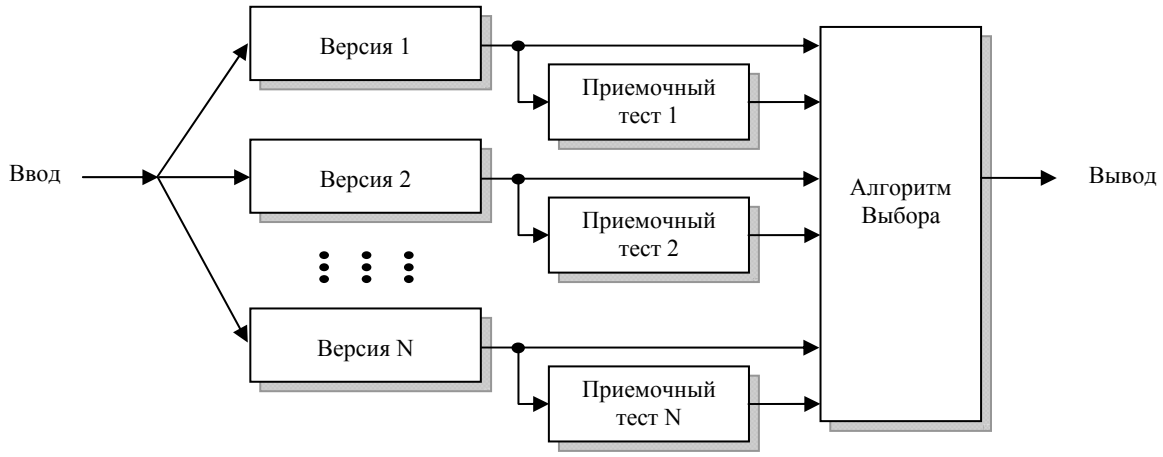

Рис. 7. Модель *N*-версионного программирования с самоконтролем

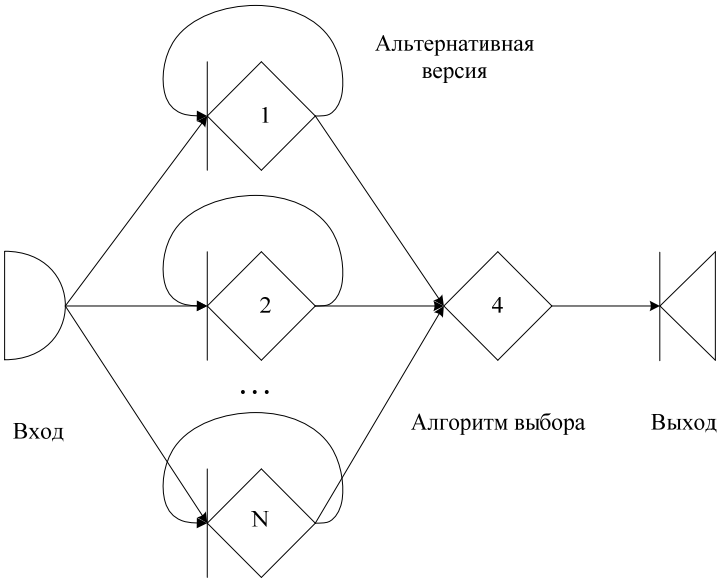

Рис. 8. Модель *N*-версионного программирования с самоконтролем в виде ГЕРТ-сети

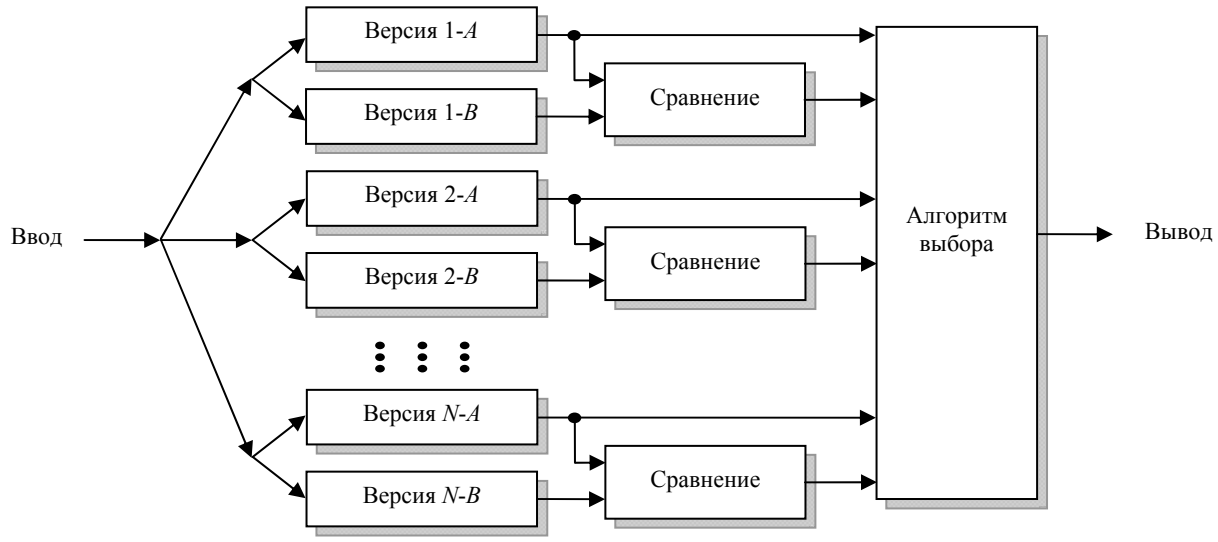

Рис. 9. Модель *N*-версионного программирования с самоконтролем, использующая сравнение

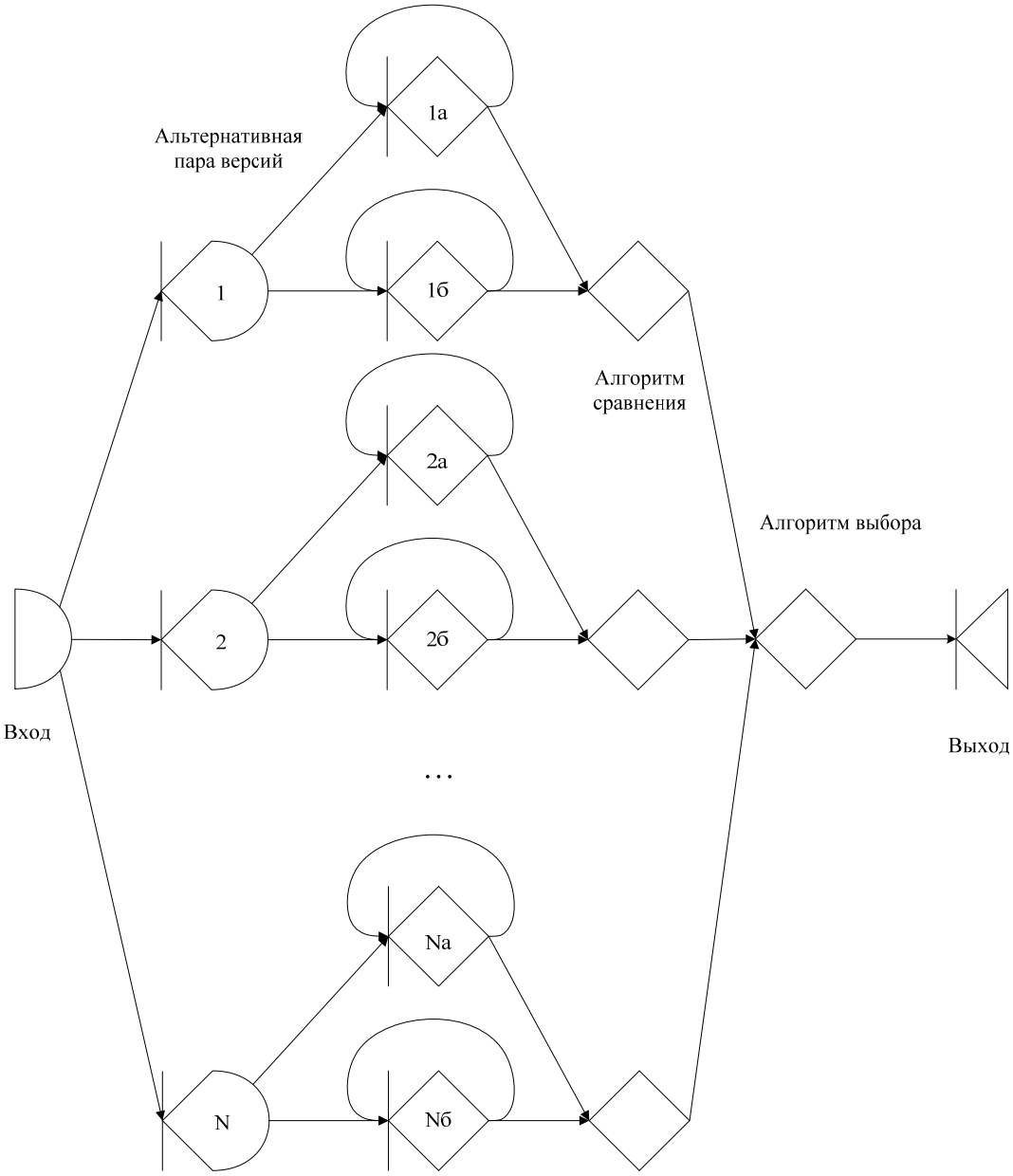

Рис. 10. Модель *N*-версионного программирования с самоконтролем, использующая сравнение в виде ГЕРТ-сети

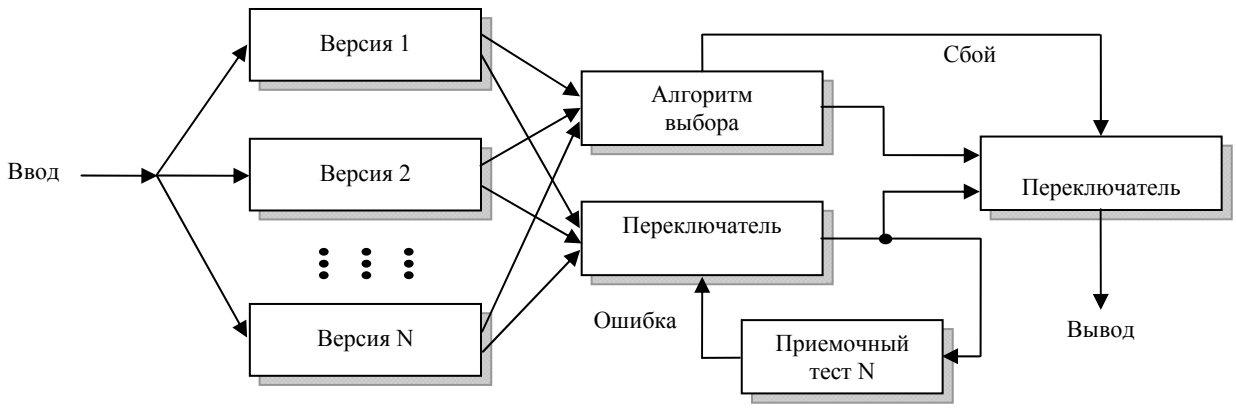

Рис. 11. Модель блока восстановления с согласованием

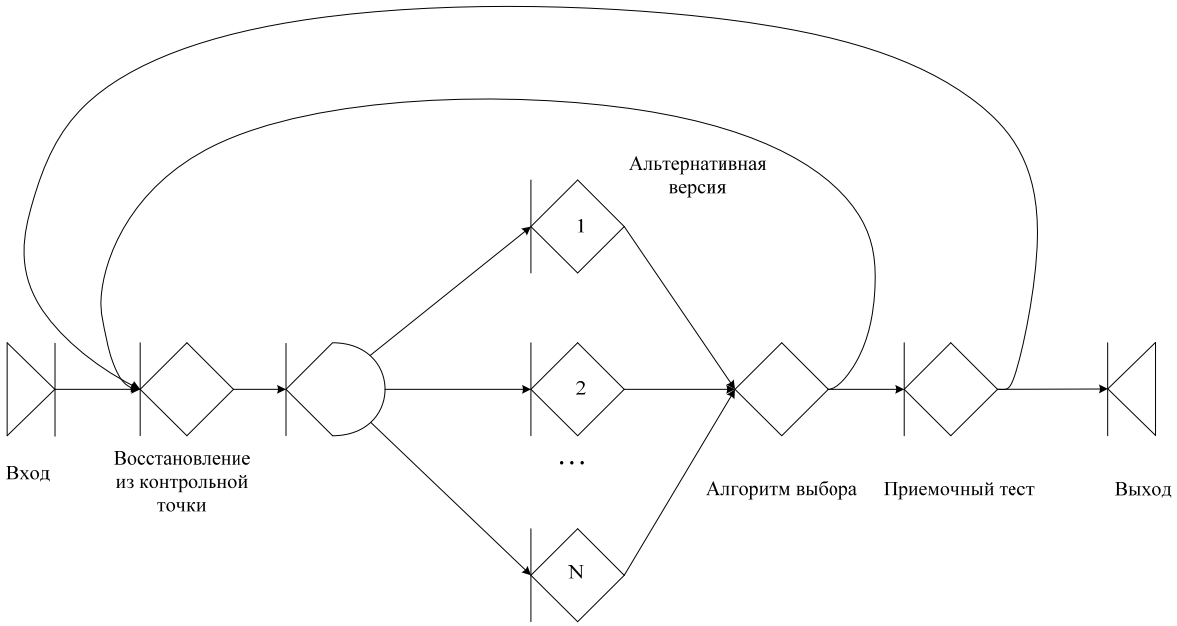

Рис. 12. Модель блока восстановления с согласованием в виде ГЕРТ-сети

Использование голосования, как в *N*-версионном программировании, не может быть применено во всех ситуациях, особенно когда возможны правильные множественные выводы. В этом случае голосование объявило бы об отказе при выборе соответствующего вывода. Блоки восстановления с согласованием используют алгоритм выбора решения, подобный *N*-версионному программированию, как первый уровень решения. Если этот первый уровень объявляет отказ, то используется второй уровень приемочного испытания, подобный тому, который был использован в методе блоков восстановления. Несмотря на очевидность гораздо более высокой сложности такого комбинированного подхода реализации, чем любого другого индивидуального метода, сравнение модели надежности указывает на то, что этот комбинированный подход имеет высокий потенциал для создания более надежного программного обеспечения. Использование слова «потенциал» важно здесь потому, что добавленная сложность реализации могла бы работать против проекта и приводить к менее надежной системе.

Предложенные способы реализации методологии мультиверсий в виде базовых ГЕРТ-сетевых моделей позволяют создавать ГЕРТ-сети, описывающие соответствующий вид мультиверсионных архитектур программного обеспечения, что делает возможным получение их вероятностно-временных характеристик функционирования. Применение необходимого и достаточного условия функционирования мультиверсионного программного модуля позволяет использовать эквивалентные преобразования и получать ГЕРТ-сети, которые могут быть рассчитаны с использованием имеющегося математического аппарата. Это существенно расширяет сферу применения ГЕРТ-сетевого моделирования для расчета вероятностно-временных характеристик мультиверсионных архитектур программного обеспечения, которые известны сейчас и могут появиться в будущем.

Библиографические ссылки

1. *Ковалев И. В., Котенок А. В.* К проблеме выбора алгоритма принятия решения в мультиверсионных системах // Информ. технологии. 2006. № 9. С. 39–44.

2. *Царев Р. Ю.* Поддержка принятия решений при мультиверсионном формировании высоконадежного программного обеспечения // Успехи современного естествознания. 2011. № 1. С. 82–83.

3. *Ковалёв П. В.* ГЕРТ-сетевой анализ мультиверсионных архитектур программного обеспечения // Успехи современного естествознания. 2009. № 9. С. 161–164.

4. *Филлипс Д., Гарсиа-Диас А.* Методы анализа сетей. М. : Мир, 1984.

5. GERT-сетевой анализ мультиверсионных программных архитектур информационно-управляющих систем / И. В. Ковалев, П. В. Ковалев, А. И. Кудинкин, Ю. А. Нургалеева // Приборы и системы. Управление, контроль, диагностика. 2010. № 8. С. 1–7.

6. Algirdas Avizienis, The Methodology of N-Version Programming. N. Y. : John Wiley & Sons, 1995.

7. *Ковалёв П. В., Лайков А. Н., Гриценко С. Н.* Определение надежности мультиверсионного программного обеспечения с использованием методов анализа сетей // Вестник СибГАУ. 2009. № 1(22). Ч. 2. С. 55–60.