УДК 004.052.3:004.032.24

ФОРМАЛИЗАЦИЯ ПРЕДСТАВЛЕНИЯ ОТКАЗОУСТОЙЧИВЫХ СИСТЕМ ПРИ ПРОЕКТИРОВАНИИ СТРУКТУРЫ СИСТЕМЫ

М. И. Глухих,
ст. преподаватель
Санкт-Петербургский государственный политехнический университет

В статье рассматривается актуальная задача синтеза структуры надежной и безопасной вычислительной системы. Рассматриваются неформальные предпосылки к определению функциональной организации таких систем. Разработана формализованная модель на основе графов, позволяющая анализировать надежностные характеристики системы на этапе системного проектирования. Применение модели рассмотрено на примере синтеза структуры простой безопасной системы

This article considers the actual task of the synthesis of reliable and safe computing system structure. Informal relations to such system functional organization definition are considered. Formalized model based on graphs analyzing system characteristics of reliability when designing the system is developed. Model's application is considered on the example of synthesis of simple safe system structure

Постановка задачи разработки структуры отказоустойчивой вычислительной системы

Сложные системы никогда не обходятся без сбоев и отказов. Как механические, так и электронные устройства постепенно изменяют свои характеристики и не всегда дают ожидаемые от них результаты. Во многих случаях наличие сбоев при работе системы допустимо, поскольку ущерб, наносимый сбоем, невелик. Однако так происходит не всегда. В некоторых случаях при проектировании систем нельзя полагаться на отсутствие сбоев и отказов. Требования к безопасности особенно жесткие в технических системах специального назначения. В некоторых случаях специализированным техническим решением вероятность отказа системы за заданное время необходимо существенно уменьшить. На решение проблемы надежности существенно влияют используемые технологии проектирования. Поэтому актуально создание отказоустойчивых систем, способных работать надежно даже при ненадежности их составных узлов.

Методы повышения надежности систем и методы *анализа* надежностных характеристик систем – довольно хорошо изученные области [1–3]. Несмотря на это, актуальной остается проблема *синтеза* вычислительной системы с высокой надежностью и безопасностью из готовых ненадежных устройств.

Учет характеристик надежности системы на этапе проектирования существенно влияет на надежность готовой системы. В связи с этим вопросы надежности следует решать одновременно с разработкой технического предложения и далее на всех последующих этапах [4].

Из-за необходимости учета большого числа факторов и неполноты информации при системном проектировании строгая методология построения задачи синтеза оптимальной структуры затруднительна. Поэтому далее развивается следующий подход. На основе неформального анализа функций, которые должна реализовать отказоустойчивая вычислительная система, определяется ее функциональная организация. При этом определяется состав типов функциональных блоков. Возможность использования различного числа экземпляров блока каждого типа и разных способов их объединения в систему порождает многообразие структур вычислительной системы.

Для сравнительного анализа структур предлагается формализованная модель и формальные правила работы с моделью. Это позволяет выделить классы наиболее целесообразных структур отказоустойчивой системы, ограничивая, таким образом, многообразие. В последующих работах будут развиты методы анализа надежностных характеристик выделенных классов систем и синтеза (через ана-

лиз) систем с заданными свойствами, учитывающих заданные ограничения на реализацию.

Постановка задачи проектирования структуры отказоустойчивой вычислительной системы предполагает рассмотрение и использование на этапе системного проектирования следующих свойств системы.

1. Отказоустойчивость. Отказ любого устройства вычислительной системы не должен приводить к остановке ее работы или, если это невозможно, количество устройств, отказ которых приводит к остановке работы системы, должно быть сведено к минимуму, т. е. система должна работать надежно на ненадежном оборудовании.

2. Достоверность результатов. Данные на выходе вычислительной системы используются таким образом, что ошибки в них могут иметь катастрофические последствия. По этой причине отказ любого количества устройств вычислительной системы не должен приводить к формированию на выходе неверных результатов или, если это невозможно, количество устройств, критических для безопасности системы, должно быть сведено к минимуму. Кратковременные остановки системы катастрофических последствий не имеют.

3. Использование готовых компонент. Часть устройств вычислительной системы (процессоры, микросхемы памяти) являются готовыми компонентами. Модификация устройств невозможна из-за закрытости их принципиальных схем. Устройства должны использоваться «как есть», надежность устройств не гарантируется.

4. Невосстанавливаемость. Остановки в ходе работы вычислительной системы устраняются путем замены вышедших из строя устройств системы на запасные исправные устройства и перезапуска системы. Неисправные устройства системы не подлежат ремонту.

Функциональная организация отказоустойчивой системы

Отказоустойчивая вычислительная система в процессе работы решает следующие задачи.

1. *Вычислительная задача*, бесперебойное решение которой является основной задачей системы. Результаты, как правило, передаются через устройства ввода-вывода системы в ее окружение.

2. *Задача передачи данных*, заключающаяся в обеспечении связи между устройствами системы.

3. *Задача синхронизации устройств*, заключающаяся в обеспечении параллельной работы устройств системы.

4. *Задача обнаружения отказов*, заключающаяся в своевременном выявлении факта их наличия в системе (конкретное место при этом не имеет значения).

5. *Задача диагностики отказов*, заключающаяся в определении конкретного устройства, в котором произошел отказ.

6. *Задача изоляции отказов*, заключающаяся в своевременном исключении неисправного устройства из системы для предотвращения процесса распространения отказа.

7. *Задача маскировки отказов*, заключающаяся в обеспечении корректных результатов на выходе системы даже при наличии отказов в отдельных ее устройствах.

В процессе проектирования отказоустойчивая вычислительная система разбивается на подсистемы, каждая из которых решает одну или более из перечисленных выше задач. Однако если набор задач, решаемых системой, достаточно очевиден, то разбиение системы на подсистемы может производиться различными способами.

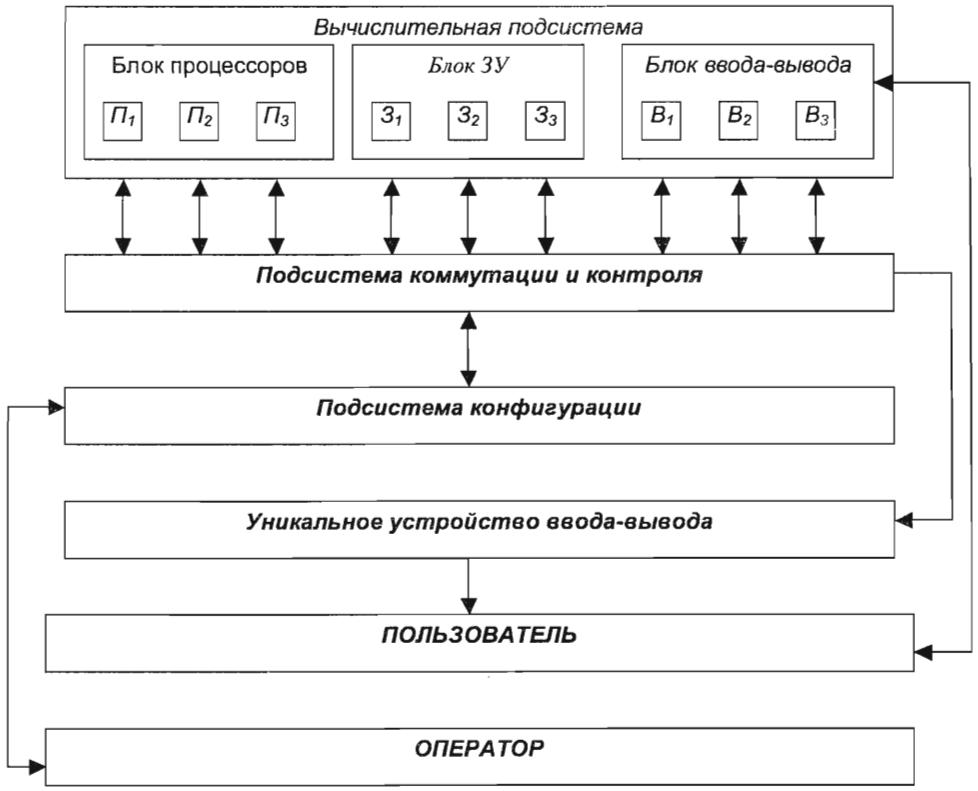
Основной подсистемой в любом случае является *вычислительная подсистема*, отвечающая за выполнение вычислительной задачи. В нее обычно входят процессоры, запоминающие устройства различного типа, устройства ввода-вывода.

Информационная связь между устройствами обеспечивается *коммутирующей подсистемой*, решающей задачу передачи данных и обычно также задачу синхронизации устройств. В нее входят магистрали и коммутаторы различных типов.

Остальные задачи специфичны для отказоустойчивой вычислительной системы. В простом случае в систему следует добавить *подсистему контроля надежности*, проверяющую корректность результатов различных устройств (например, путем сравнения их друг с другом) и решающую задачи обнаружения, маскировки и первичной диагностики отказов. Однако задача передачи данных от устройства к устройству предполагает также параллельную проверку корректности данных. В связи с этим, удобно объединить коммутирующую подсистему и подсистему контроля надежности в единую *подсистему коммутации и контроля*. Эта же подсистема, используя собранную информацию об отказах отдельных устройств, может решать задачу изоляции отказов.

Кроме этого, в систему добавляется *подсистема конфигурации*, собирающая информацию о сбоях и отказах от подсистемы контроля надежности. Подсистема конфигурации отображает эту информацию оператору и обеспечивает возможность ручного изменения конфигурации системы и проведение ее тестирования. Таким образом, подсистема конфигурации решает задачи восстановления системы и вторичной диагностики отказов.

Основными элементами окружения системы являются пользователь и оператор [5]. Пользователь – основной потребитель системы – связывается с системой через устройства ввода-вывода. Вид этих устройств зависит от класса решаемых задач – это может быть, например, монитор, отображающий результаты анализа информации, или двигатель, управляемый по некоторому алгоритму. Однако, в любом случае, некоторые устройства ввода-вывода должны присутствовать в системе в единственном экземпляре – решение о корректности тех или иных результатов должно приниматься внутри отказоустойчивой вычислительной системы. Такие устройства ввода-вывода будем называть *уникальными*.



■ ***Рис. 1.*** *Функциональная схема отказоустойчивой системы и ее окружения*

Оператор осуществляет замену неисправных устройств на исправные, проводит тестирование системы и ручную настройку ее конфигурации, если это требуется. Таким образом, оператор связан с подсистемой конфигурации.

Подобное распределение задач приводит к функциональной схеме системы, представленной на рис. 1.

Формализованная модель отказоустойчивых систем

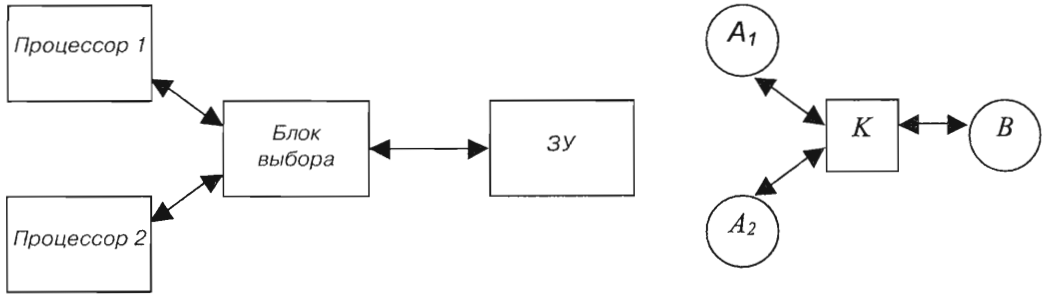
Основная задача следующего этапа синтеза системы – определение состава и количества блоков в вычислительной подсистеме и разработка структуры подсистемы коммутации и контроля надежности.

Пусть имеется вычислительная подсистема, состоящая из процессора, запоминающего устройства, устройств ввода-вывода и других устройств. Внутренняя структура подобных устройств для изменения недоступна, и все или некоторые из них не являются абсолютно надежными. Задача состоит в следующем. Используя, возможно, по несколько экземпляров каждого из устройств, соединить их с помощью системы коммутации и контроля в отказоустойчивую вычислительную систему. Подсистема коммутации и контроля не является готовой и подлежит разработке. На этапе системного проектирования при разработке структуры вычислительной системы для подсистемы коммутации и контроля определяются функциональные спецификации.

При синтезе и сравнительном анализе структур отказоустойчивой системы возможно абстрагирование от конкретной функциональности вычислительных устройств и набора передаваемых между ними сигналов. Интерес представляют виды устройств, используемых в системе, а также наличие либо отсутствие связей между ними. Адекватным аппаратом для такого абстрактного представления систем является теория графов [6, 7].

Основные функции устройств коммутации и контроля – это передача данных от входных устройств к выходным, контроль корректности результатов двух одинаковых устройств путем сравнения, определение двух исправных устройств из трех путем голосования, фиксация неисправности входного устройства путем набора статистики и т. д. Заменим устройства системы вершинами графа, а связи между ними – направленными дугами графа. Например, рассмотрим систему, в состав которой входят два процессора и одно запоминающее устройство.

Граф однозначно определяет структуру системы. Для контроля текущего состояния системы введем переменные *состояния устройств* (здесь имеется в виду фактическое состояние устройства, оно не всегда известно другим устройствам системы и вне системы). Изначально все устройства находятся в *рабочем состоянии W*. Если извне поступает команда остановки, либо устройство замечает внутреннюю ошибку (если у него есть такая возможность), оно переходит в *состояние остановки S*.



■ ***Рис. 2.*** *Преобразование структуры простой системы в граф*

Если же на вход устройства поступают неправильные данные извне либо происходит внутренний незамеченный отказ, искажающий выходные данные, оно переходит в *состояние аварии F*. В последнем случае само устройство не может определить некорректность своего состояния. В большинстве случаев авария устройства диагностируется одним или несколькими устройствами коммутации и контроля.

Устройство подсистемы коммутации и контроля, на входы которого поступает несколько результатов от одинаковых устройств, в случае поступления неправильных данных на один из входов останавливается, если сравнивалось два результата, или продолжает работать, если результатов было три или более. Например, в системе, представленной на рис. 2, отказ ЗУ приводит к переходу блока выбора в состояние *F*, отказ процессора *1* приводит к переходу блока выбора в состояние *S*. Если добавить в систему процессор *3*, отказ процессора *1* не приведет к изменению состояния блока выбора – правильные данные будут установлены мажорированием.

Отказ любого устройства системы может привести к изменению состояния системы в целом. Система находится в рабочем состоянии *W*, если в ней присутствует хотя бы минимальный набор работающих устройств, связанных друг с другом. Требования к минимальному набору определяются в дальнейшем.

Если это условие не выполняется, система переходит в состояние остановки *S* или аварии *F*. Критерий остановки – переход хотя бы одного устройства системы в состояние *S*. При этом считается, что некорректное состояние системы диагностировано одним из устройств коммутации и контроля, перешедшим в состояние *S*.

В противном случае система переходит в состояние аварии. Состояние аварии системы может быть диагностировано только устройствами, внешними по отношению к системе.

Для работоспособности системы, приведенной на рис. 2, требуется, чтобы все четыре ее устройства находились в состоянии *W*. Отказ любого процессора приводит к остановке системы, отказ других устройств приводит к аварии.

Для формализации сравнительного анализа структур отказоустойчивой системы зададим систему правил представления системы графом и правила его преобразования при изменении состояния устройств системы.

Система правил построения и работы с графом отказоустойчивой системы[1]

Правила представления системы в виде графа (группа I).

1. Система может быть представлена в виде изоморфного ей ориентированного графа [6], при этом каждое устройство системы будет изоморфно одной из вершин графа, а каждая связь системы будет изоморфна одной из дуг графа. Для любой пары вершин (A, B) существует не более одной дуги $A \rightarrow B$ (с учетом направления дуги). Если существует дуга $A \rightarrow B$, в дальнейшем будем говорить, что вершина A – входная для вершины B, а вершина B – выходной для вершины A. В графе отсутствуют петли $A \rightarrow A$ для любой вершины A. В дальнейшем изоморфный данной системе граф будет называться графом системы. *Все потоки данных от устройства A к устройству B обозначаются одной дугой. Потоки данных внутри одного устройства на графе не отображаются.*

2. Множество вершин графа делится на два непересекающихся подмножества: подмножество основных вершин и подмножество промежуточных вершин. Устройства вычислительной подсистемы представляются основными вершинами (на рисунках они круглые), устройства подсистемы коммутации и контроля – промежуточными вершинами (на рисунках они квадратные). Подсистема конфигурации на графе не отображается. Следует учитывать, что задача передачи информации в системе решается устройствами коммутации и контроля, поэтому возможные искажения данных происходят в промежуточных вершинах, а не в дугах графа системы. Дуги при этом считаются идеально надежными, так как отображают не линии передачи информации, а подключение одного устройства системы к другому.

Правила соединения вершин графа системы (группа II).

1. Основные вершины в графе могут быть связаны только через одну или несколько промежуточных

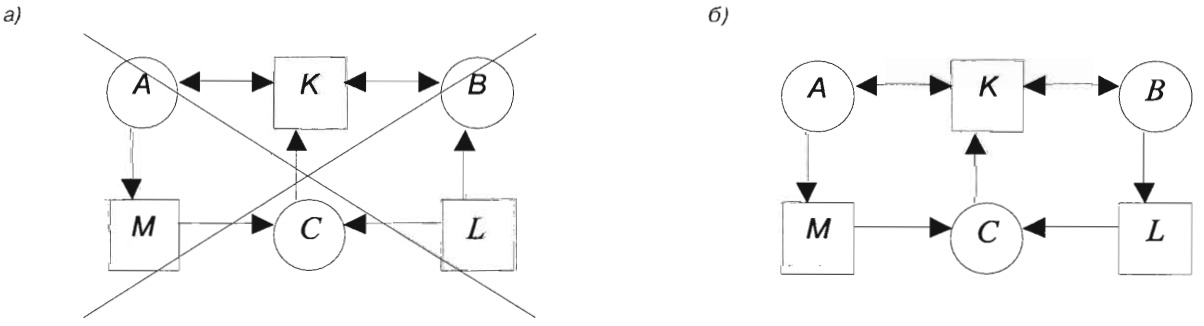
[1] Прямым шрифтом записан непосредственно текст правил, курсивом – комментарии и примеры к нему.

вершин. Множество основных вершин, не связанных непосредственно, называют независимым, или внутренне устойчивым [6, 7]. Множество промежуточных вершин при этом называют доминирующим, или внешне устойчивым [6, 7]. *Правило отражает тот факт, что задачу передачи информации решают промежуточные вершины (см. правило I.2 и рис. 3).*

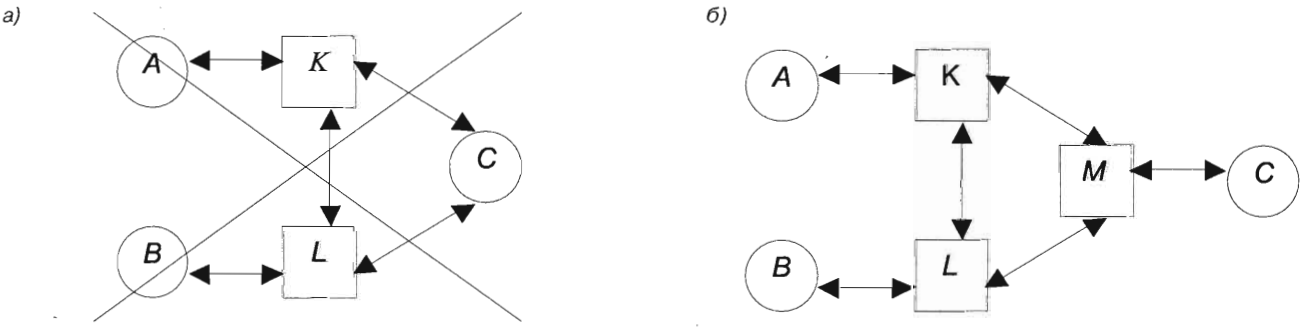
2. Для обеспечения информационного взаимодействия вычислительных устройств в системе любые две основные вершины графа A и B должны быть взаимодостижимыми [6, 7], т. е. существует цепь $A \to C_1 \to C_2 \to \ldots \to C_n \to B$ и цепь $B \to D_1 \to D_2 \to \ldots \to D_m \to A$. Любая промежуточная вершина должна использоваться для передачи информации, т. е. для промежуточной вершины K должны существовать две основные вершины A и B, такие, что существует цепь $A \to C_1 \to \ldots \to C_n \to K \to D_1 \to \ldots \to D_n \to B$ (иначе соответствующее вершине устройство бесполезно и может быть удалено). Из этих двух требований следует сильносвязность любого графа системы (все основные вершины связаны друг с другом, а все промежуточные связаны по входу и выходу хотя бы с одной основной). *Правило отражает возможность передачи информации между любыми двумя устройствами системы. На рис. 4, а, в графе слева нет ни одной цепи, ведущей в вершину L – граф является слабосвязным. Изменив направление связи $L \to B$, получим сильносвязный граф (рис. 4, б).*

3. У всех вычислительных устройств входы присутствуют в одном экземпляре (устройства имеют одну входную шину); коммутация данных, поступивших от различных вычислительных устройств, происходит в подсистеме коммутации и контроля. Поэтому все основные вершины графа могут иметь только одну входящую дугу. Этого можно достигнуть следующим преобразованием графа: все основные вершины A, в которые входит более одной дуги, расщепляются на пару $F \to B$, после чего все входные вершины для A становятся входными для F, а все выходные вершины для A становятся выходными для B. Промежуточные вершины, входные для основных вершин, в дальнейшем называются формирующи-

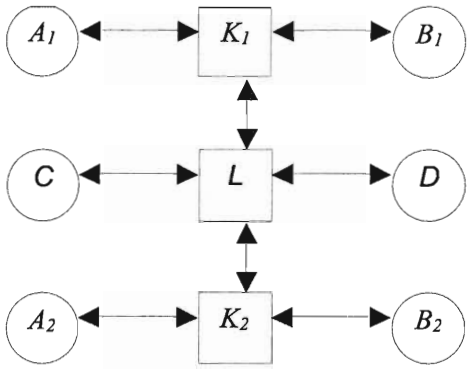
■ **Рис. 3.** *Неправильная (а) и правильная (б) связь основных вершин*



■ **Рис. 4.** *Пример слабосвязного (а) и сильносвязного (б) графа системы*



■ **Рис. 5.** *Неправильная (а) и правильная (б) конфигурация промежуточных вершин*



■ ***Рис. 6.*** *Промежуточные вершины одного типа (K_1, K_2) и другого типа (L)*

ми вершинами. *На рис. 5, а, у основной вершины C две входных вершины K и L, что ошибочно. Для устранения ошибки в граф добавляется промежуточная вершина M – формирующая для вершины C (рис. 5, б).*

Правила введения типов вершин (группа III).

1. В отказоустойчивой вычислительной системе существуют устройства, имеющие одинаковую внутреннюю структуру и выполняющие идентичные операции во время работы системы. Про такие устройства будем говорить, что они имеют один и тот же тип, и сопоставим им вершины графа системы, также имеющие один и тот же тип. *На рисунках вершины одного типа имеют одинаковую первую букву в названии.*

2. Основная вершина *A* и промежуточная вершина *K* всегда имеют различные типы, так как изоморфные им устройства всегда выполняют в системе различные операции.

3. Устройства коммутации и контроля можно считать выполняющими идентичные операции в том и только в том случае, когда у них совпадают как входные, так и выходные потоки данных. Входные потоки данных совпадают, если формируются устройствами одного типа. Выходные потоки данных совпадают, если принимаются устройствами одного типа. По этой причине тип промежуточных вершин инвариантен относительно любого автоморфизма графа, сохраняющего множества основных и промежуточных вершин графа, а также типы основных вершин графа (назовем такой автоморфизм системным). Автоморфизм графа – взаимно однозначное преобразование его вершин, сохраняющее все отношения смежности «входная вершина – выходная вершина» [6, 7]. Иначе говоря, промежуточные вершины *A* и *B* относятся к одному типу тогда и только тогда, когда существует системный автоморфизм, ставящий их в соответствие. *На рис. 6 вершины K_1 и K_2 имеют одинаковый тип, а вершина L имеет другой тип. Преобразование $A_1 \leftarrow\rightarrow A_2$, $B_1 \leftarrow\rightarrow B_2$, $K_1 \leftarrow\rightarrow K_2$ сохраняет все отношения смежности, множества вершин и типы основных вершин и ставит в соответствие вершины K_1 и K_2.*

Следствия из группы III.

1. Если промежуточные вершины K_1 и K_2 принадлежат к одному типу и вершина L_1 является входной для K_1, то существует вершина L_2 – входная для K_2, тип которой совпадает с L_1 (возможно, вершины L_2 и L_1 совпадают). *Следует из правила III.3.*

2. Если промежуточные вершины K_1 и K_2 принадлежат к одному типу, и вершина L_1 является выходной для K_1, то существует вершина L_2 – выходная для K_2, тип которой совпадает с L_1 (возможно, вершины L_2 и L_1 совпадают). *Следует из правила III.3.*

3. Если основные вершины A_1 и A_2 принадлежат к одному типу, то их формирующие вершины F_1 и F_2 также принадлежат к одному типу. *Аналог следствия 1 для основных вершин. Отметим, что аналог следствия 2 для основных вершин может и не выполняться.*

Правила изменения состояний вершин (группа IV).

1. Любая вершина всегда находится в одном из трех состояний: работа *W* (Work), остановка *S* (Stop), авария *F* (Fail). *Состояния устройств системы представляются состояниями вершин графа. Напомним, что здесь имеется в виду фактическое состояние устройства.*

2. Состояние вершин графа меняется с течением времени. В начальный момент времени все вершины графа находятся в состоянии *W*. *Правило отображает момент начального пуска системы, когда она находится в полной конфигурации.*

3. Вершина, однажды оказавшаяся в состоянии *S* или *F*, в последующие моменты времени не изменяет свое состояние. *Как было сказано выше, возможность восстановления системы не рассматривается.*

4. В любой вершине *A*, находящейся в состоянии *W*, в произвольный момент времени может произойти событие, называемое отказом, причем отказ может быть обнаруженным или нет. Если происходит обнаруженный отказ, вершина переходит в состояние *S*. Если происходит необнаруженный отказ, вершина переходит в состояние *F*. *Правило отображает внутренние отказы устройств системы. Отказ будет обнаруженным, если о нем сообщит внутренняя система контроля данного устройства. В дальнейшем, если не указано противное, отказ считается необнаруженным.*

5. Изменение состояния вершины *A* может повлечь за собой изменение состояния вершины *B*, находящейся в состоянии *W* и являющейся выходной для вершины *A*. Если состояние вершины *B* меняется – это происходит в тот же момент времени, когда изменилось состояние вершины *A*. *Правило отображает возможность распространения отказа по цепи.*

Правила распространения отказа (группа V).

1. Изменение состояния любой формирующей вершины *F* приводит к тому, что все основные вер-

шины A, выходные для вершины F, переходят в то же самое состояние. *Это означает, что остановившееся устройство коммутации и контроля отправляет сигнал остановки во все связанные с ним вычислительные устройства, а отказавшее устройство коммутации и контроля передает на вход всем связанным с ним вычислительным устройствам неверные данные.*

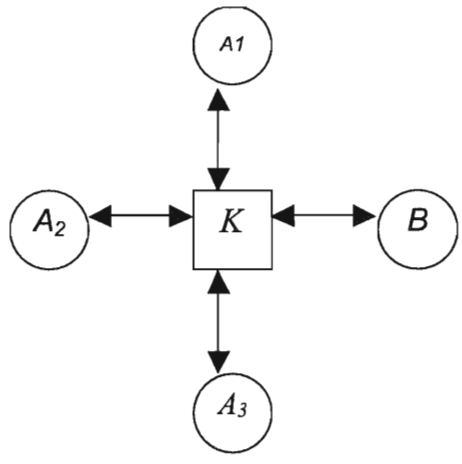
2. Пусть промежуточная вершина K находится в состоянии W, а входная для нее вершина A_i изменяет свое состояние. Тогда последующее состояние вершины K будет зависеть от количества вершин A_j, входных для вершины K, находящихся в состоянии W и принадлежащих к одному типу с вершиной A_i.

Если не существует ни одной вершины A_j, вершина K переходит в то же состояние, в которое перешла вершина A_i, ситуация с отсутствием резерва, остановившееся устройство останавливает и все свои выходные устройства, отказавшее устройство передает им на вход неверные данные.

Если существует ровно одна вершина A_j, вершина K переходит в состояние S, контроль сравнением, данные, поступившие на вход от двух идентичных устройств, не совпадают, устройство останавливает свою работу.

Если существует более одной вершины A_j, вершина K остается в состоянии W, контроль голосованием, данные, поступившие на вход от одного из устройств, не совпадают с результатами двух других. Входное устройство признается отказавшим, устройство коммутации и контроля продолжает свою работу в урезанной конфигурации.

На рис. 7 отказ вершины B приводит к переходу вершины K в состояние F (отсутствие контроля). Отказ любой одной вершины типа A оставляет вершину K в состоянии W (контроль голосованием, система продолжает работу в неполной конфигурации с двумя вершинами типа A). Отказ второй вершины типа A переводит вершину K в состояние S (контроль сравнением).



■ ***Рис. 7.*** *Иллюстрация распространения отказа в промежуточную вершину*

Правила изменения состояния графа (группа VI).

1. Граф находится в одном из трех состояний: работа W, остановка S, авария F. *«Состояние графа» соответствует состоянию системы, изоморфной ему.*

2. Состояние графа не меняется, если не меняются состояния его вершин.

3. Граф, однажды оказавшийся в состоянии S или F, не меняет свое состояние.

4. Изменение состояния вершин графа может повлечь за собой переход графа из состояния W в состояние S или F.

5. Граф находится в состоянии W, если и только если существует сильносвязный подграф из вершин, находящихся в состоянии W, среди которых имеется хотя бы одна основная вершина каждого типа. *Пока в системе остается работающее вычислительное устройство каждого типа, система функционирует.*

6. Пусть граф находился в состоянии W и произошел отказ одной из его вершин. Если граф перестал находиться в состоянии W (условие правила 5 не выполняется) и хотя бы одна основная вершина, непосредственно перед отказом находившаяся в состоянии W, перешла в состояние S, граф также переходит в состояние S; в обратном случае граф переходит в состояние F. *Правило задает условие безопасности отказа системы. На рис. 7 отказ вершины B или K переводит граф в состояние F, отказ одной вершины типа A оставляет граф в состоянии W, отказ второй вершины типа A переводит граф в состояние S.*

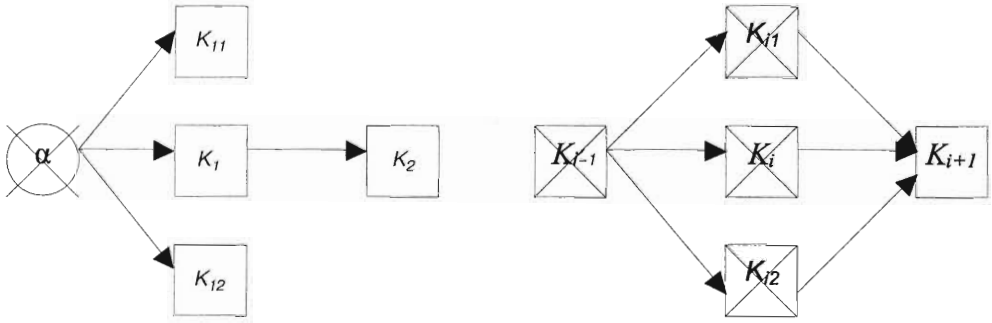
Условия безопасности систем

Очевидно, что для любого графа системы существуют последовательности отказов, выводящие граф из состояния W – ведь после отказа всех вершин графа, по правилу VI.5, он не может находиться в состоянии W. Для обеспечения безопасности ситуация несколько иная. Можно построить граф так, чтобы любая последовательность отказов оставляла его в состоянии W или приводила его в состояние S, но не могла приводить к состоянию F. Систему, граф которой обладает таким свойством, назовем безопасной. На практике это означает, что любая последовательность отказов устройств в системе не может привести к аварии – к ней может привести только одновременный отказ хотя бы двух устройств, и то не всегда. Одновременные отказы при дальнейшем рассмотрении не учитываются, так как они происходят гораздо реже последовательных.

Для иллюстрации применения модели рассмотрим требования к безопасным системам.

Отсутствие решающих вершин. Будем называть некоторую вершину графа решающей, если в графе не существует другой вершины, тип которой совпадает с данной. Справедливо следующее утверждение.

Пусть в графе существует цепочка $\alpha \rightarrow K_1 \rightarrow \ldots \rightarrow K_n$, α – решающая вершина, $K_1, \ldots, K_n$ – произволь-



■ ***Рис. 8.*** *Индуктивная база и индуктивный переход*

ные основные или промежуточные вершины. Тогда переход вершины α в состояние *F* вызовет переход в состояние *F* всех вершин из этой цепочки.

Докажем по индукции, что все вершины цепочки перейдут в состояние *F*.

Б а з а. Поскольку для вершины K_1 нет второй входной вершины, тип которой совпадает с α (вершина α – решающая), по правилу V.2 вершина K_1 перейдет также в состояние *F*. Кроме того, если в графе есть вершины, тип которых совпадает с K_1, они также перейдут в состояние *F*, поскольку по следствию III.1 вершина α должна быть для них входной. Таким образом, переход вершины α в состояние *F* вызывает переход всех узлов типа K_1 в состояние *F*.

П е р е х о д. Пусть переход всех вершин типа K_i – –1 в состояние *F* вызывает переход всех вершин типа K_i в состояние *F*. У вершины K_i + 1 (или вершины, тип которой совпадает с K_i +1) может быть несколько входных вершин типа K_i, однако все они перейдут в состояние *F* – по индуктивному предположению. Поэтому все вершины типа K_i +1 также перейдут в состояние *F* – индуктивный переход доказан. Лемма доказана (рис. 8).

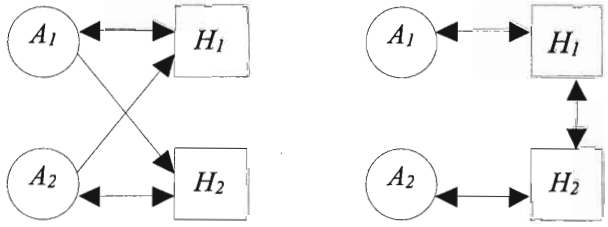
Так как граф системы является сильносвязным (правило II.2), для любой вершины *X* существует цепочка $\alpha \to L_1 \to \ldots \to LN \to X$. Отсюда следует, что отказ вершины α вызывает переход всех вершин графа в состояние *F* и, как следствие (правило VI.6), переход графа в состояние *F*. Поэтому в графе безопасной системы не может быть решающих вершин.

Наличие дублированного контроля. Если граф некоторой системы безопасен, то выполняется одно из двух следующих условий.

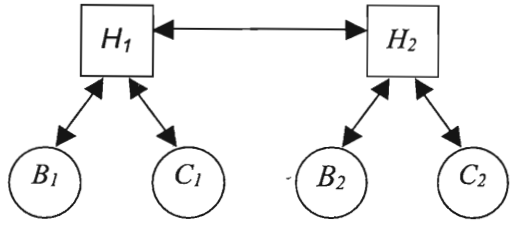
1. В графе присутствуют две промежуточных вершины одного типа K_1 и K_2, и у каждой из них есть входная промежуточная вершина такого же типа (в том числе, они могут быть входными друг для друга).

2. В графе присутствуют две промежуточных вершины *L* и *M* (необязательно одного типа), и у каждой из них имеется как минимум две входных вершины одного типа1.

Устройства коммутации и контроля также могут выходить из строя. Для безопасности системы необходимо, чтобы устройства коммутации и контроля были дублированы, и чтобы каждое из них осуществляло сравнение результатов – либо двух вычислительных устройств (2), либо своих и такого же устройства коммутации и контроля (1).



■ ***Рис. 9.*** *Варианты графа надежной и безопасной системы с парой основных вершин одного типа*



■ ***Рис. 10.*** *Безопасный граф с двумя парами основных вершин*

Докажем это утверждение. В безопасном графе должна существовать некоторая вершина, которая при переходе в состояние *F* одной из своих входных вершин сама не перейдет в состояние *F* (если это условие не выполнено, отказ любой вершины автоматически приводит к переходу всех вершин системы в состояние *F*). Возможны два варианта такой вершины.

1. У вершины K_1 имеется входная вершина K_2 такого же типа. Автоматически это означает, что у вершины K_2 есть входная вершина того же типа, что и K_1 (следствие III.1).

2. У вершины *L* имеются две входных вершины одного типа. Если такая вершина единственна, ее отказ автоматически приведет к переходу в состояние *F* всех остальных вершин системы – а значит, система не безопасна. Поэтому должна иметься вторая вершина *M* с двумя входными вершинами одного типа, либо должно быть выполнено условие 1.

Для графа системы с парой основных вершин одного типа два этих условия приводят к двум наиболее простым структурам (рис. 9).

Граф надежной и безопасной системы с двумя и более парами основных вершин строится аналогич-

но. Поскольку в рассмотренных структурах отказ любого устройства приведет как минимум к остановке системы, достаточно присутствия в графе двух промежуточных вершин. Возможная структура изображена на рис. 10.

Заключение

Таким образом, применив построенную модель, мы получили ответ на вопрос – как построить безопасную систему наиболее простым способом. Для синтеза с помощью построенной модели системы с наиболее надежной или безопасной структурой модель необходимо дополнить методикой оценки надежности и безопасности системы по характеристикам отдельных ее устройств. В частности, с ее помощью можно доказать, что приведенные на рис. 9, 10 структуры являются наиболее дешевыми и безопасными для заданного набора основных вершин, как будет показано в дальнейших исследованиях.

Литература

1. **Пирс У.** Построение надежных вычислительных машин. – М.: Мир, 1968. – 270 с.
2. **Рябинин И. А.** Надежность и безопасность структурно-сложных систем. – СПб.: Политехника, 2000. – 248 с.
3. **Черкесов Г. Н.** Надежность аппаратно-программных комплексов. – СПб.: Питер, 2005. – 480 с.
4. **Нечипоренко В. И.** Структурный анализ систем. – М.: Советское радио, 1977. – 216 с.
5. **Heimendinger W., Weinstock C.** A conceptual framework for system fault tolerance. – Pittsburgh (PA): Carnegie Melon University, 1992. – 36 p.
6. **Алгоритмы** и программы решения задач на графах и сетях / Под ред. М. И. Нечепуренко – Новосибирск: Наука, 1990. – 520 с.
7. **Оре О.** Теория графов. – М.: Наука, 1980. – 336 с.