МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ

УДК 519.21

А.С.Сорокин

СТРУКТУРНОЕ МОДЕЛИРОВАНИЕ НАДЕЖНОСТИ ТЕХНОЛОГИЧЕСКИХ СИСТЕМ С ИСПОЛЬЗОВАНИЕМ СКЕЛЕТОНОВ

Введение. Проектирование технологических систем с использованием высокоуровневых конструкций имеет явные преимущества. Это известно в структурном параллельном и распределенном программировании, где часто последовательные подзадачи структурируют до параллельного узла [1-4]. Технический план решения вслед за этими видами описания имеет своей целью получение высокоуровневого описания структуры вычислений, которое может быть легко переформировано для определения хорошей привязки задач к вычислительным ресурсам. Пользователь заинтересован в достижении соответствующей производительности операций до тех пор, пока не будут удовлетворены ограничения по использованию элементов среды выполнения.

Тем не менее, хотя композиционные языки и позволяют компоновать подзадачи, они не обеспечивают механизм оценки того, превосходит ли новая версия данной модели применяемая ранее.

Кроме того, языки, подходящие для моделирования технологических систем, такие как алгебры стохастических процессов, обычно не структурированы в данном смысле, и не имеют лингвистического аппарата для выражения композиции подзадач. Целью данной работы является ликвидация этого разрыва путем автоматической генерации моделей алгебры процессов из структурных описаний приложения. Это таким образом, позволит пользователю компилировать свои технические приложения в моделях алгебры процессов, подходящих для оценки надёжности технологических систем посредством анализа детерминированной характеристики или анализа переходных процессов для адекватности модели по вероятности. Свое внимание сосредоточим на результатах, которые могут быть получены посредством марковского стационарного анализа моделей алгебры процессов [5-7]. Будем использовать алгоритмические скелетоны [8] как пример структурного композиционного языка. В качестве алгебры процессов используется стохастическая алгебра процессов для оценки надёжности.

1.Теоретические основы. Представим краткое описание используемой стохастической алгебры процессов и метода структурного параллельного и распределенного программирования, основанного на алгоритмических скелетонах [8-10].

1.1 Стохастическая алгебра процессов для оценки надёжности. PEPA – это алгебра марковских процессов. В ней каждому действию сопоставляется экспоненциально распределенная случайная переменная, представляющая собой длительность процесса технологии. Модели составляются из *компонентов*, которые взаимодействуют посредством *действий*. Синтаксис языка следующий:

$$S ::= (\alpha, r).S \mid S + S \mid C_s$$

(префикс, селекция и имя компонента)

$$P ::= P \rhd_L \lhd P \mid P / L \mid C$$

(кооперация, абстракция и имя компонента)

Здесь S обозначает *последовательный компонент,* а P - моделируемый компонент, который выполняется параллельно. C означает константу, обозначающую либо последовательный компонент, либо модельный компонент как представлено определением. C_s означает константы, определяющие последовательные компоненты. Префикс $(\alpha,r).S$ задает компоненту назначенное первым действие: он будет иметь действие типа α и продолжительность, определяемую экспоненциальным распределением с параметром r. Оператор селекции $(+)$ разрешает действия его двух операндов. Завершенное первым действие отличает одно из них: другое не учитывается. Система будет вести себя как производный результат развития выбранного компонента. Структура представляется моделью, использующей комбинатор кооперации $(\rhd_L \lhd)$. В PEPA больше нет дополнительных действий и это заимствовано из CSP-подхода параллельного составления: компоненты синхронизируются по тем действиям во множестве кооперации L, которые разрешены обоими компонентами. Синхронизация ограниченной мощности, означает, что синхронизация не может ускорить выполнение компонента. Таким образом, длительность общих действий определяется минимумом случайных переменных, соотносимых с каждым из участвующих дейст-

вий. В некоторых случаях компонент пассивен по отношению к синхронизируемому действию, что означает, что он будет участвовать в той мере, в какой ожидается его компонентом-партнером. Это действие обозначается специальным символом $\perp$, т.е. $(\alpha, \perp)$. Когда множество кооперации L пусто, то используется условное обозначение 2. Операция абстракции, обозначаемая *P/L* , позволяющая типам действий, чей тип находится в L, быть замененными специальным типом τ, который обозначает закрытое или скрытое действие.

Описание языка представлено в источнике [9] в виде структурированной на мелкие шаги операционной семантики, которая отображает модели PEPA на марковский процесс с непрерывным временем.

1.2 Понятие алгоритмических скелетонов. Скелетонный подход для проектирования техноло-гических параллельных систем предполагает, что сложность параллельного программирования может быть ограничена путем сдерживания в определен-ных пределах подсистем, посредством которых может проявляться параллелизм. Ограничение сложности достигается путем сведения ее к небольшому числу архитектурно независимых конструкций, известных как “алгоритмические скелетоны”. Каждая спецификация скелетона включает в себя “логическое поведение” схем параллельного вычисления, в то время как детали его конкретной реализации компонуются и скрываются. Обеспечение скелетонно-ориентированной методологии программирования одновременно повышает уровень абстракции, которым оперирует пользователь и обеспечивает систему планирования ключевой информации о временных и структурных взаимодействующих схемах. Ответственность за использование этой информации переходит от пользователя к компилятору или к системе поддержки исполнения программ. Получение такой детальной информации из эквивалентной программы, передающей специализированные $(ad-hoc)$ сообщения, в общем случае невыполнимо. В работе показываем, каким образом структурная информация может быть использована для формирования моделей производительности приложения в PEPA.

2. Представление структурированных приложений с помощью скелетонов. Для автоматического формирования моделей надёжности заданное приложение сначала должно быть представлено в форме, выражающей его сущность.

В данной работе адаптируется схемно-ориентированный подход, основанный на понятии алгоритмических скелетонов – системы, которая была разработана для улучшения и упрощения структурной разработки распределенных и параллельных приложений. Если представленные конструкции могут использоваться непосредственно пользователем модели, они предназначены для применения в качестве внутреннего интерфейса и для автоматического формирования моделей надёжности из базирующихся на скелетонах распределенных приложений. Для упрощения детального рассмотрения автоматизации сосредоточим внимание на следующих трех базовых скелетонах.

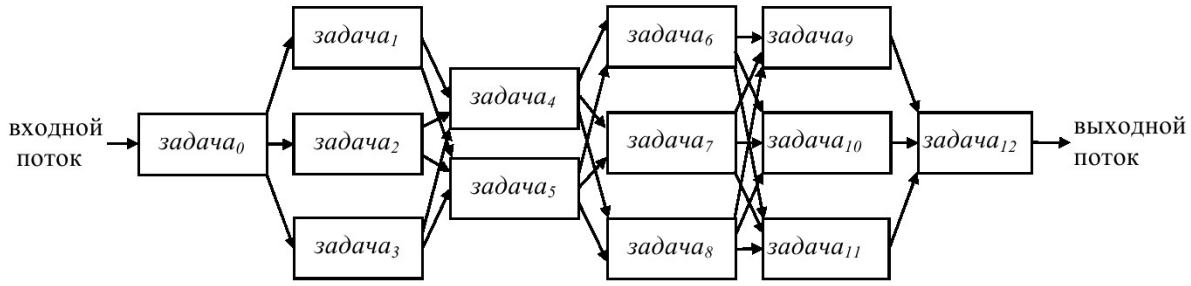


Рис. 1. Схема потоков данных приложения с вложенными Pipeline и дублированиями задач.

Скелетон Pipeline (Канал). Скелетон Pipeline располагает множество компонентов последовательно, так что элементы данных, поступая в Pipeline, обрабатываются в каждом из этих компонентов по очереди (в том порядке, в каком компоненты определены) перед тем как конечный результат покидает Pipeline. Для определения данного скелетона будем использовать следующую конструкцию:

pipe(<*количество компонентов***>); .**

Компоненты, входящие в скелетонную конструкцию, могут быть как скелетонными компонентами (результаты которых иерархически вложены), или заданными компонентами (в которых элементы данных обрабатываются). Заданный компонент определяется следующей конструкцией:

task(<*имя компонента***>, <***интенсивность***>); .**

Здесь **<***интенсивность***>** - это интенсивность, с которой обрабатывается каждый из элементов данных, поступающих в заданный компонент. Она используется в процессе моделирования вычислительной производительности задачи.

Скелетон Deal (Распределитель). Скелетон Deal дублирует данный заданный компонент параллельно, так что элементы данных, поступая в Deal, могут быть обработаны одним из дублирующих компонентов, до того, как конечный ре-

зультат покидает Deal. Заданный компонент, который получает заданный элемент данных, выбирается согласно правилу кругового распределения данных. Для определения данного скелетона используем следующую конструкцию:

deal(<*число дублирований*>, <*имя компонента*>, <*интенсивность*>**); .**

Скелетон Farm (Ферма). Скелетон Farm похож на Deal: данный задачный компонент дублируется параллельно, так что элементы данных, поступая в Farm, обрабатываются одним из компонентов. Отличие в том, что процесс выбора задачного компонента, который должен получить заданный элемент данных, невозможно прогнозировать, т.к. он динамически управляем более ранними вычислениями. В этом случае следующий элемент данных посылается одной из задач-преемников, завершающих обработку ранее назначенных ей элементов данных. Для определения Farm используем конструкцию вида:

farm(<*число дублирований*>, <*имя компонента*>, <*интенсивность*>**); .**

Рассмотрим конкретный пример применения этих конструкций. Представим технологическую систе-му, подобную той, которая изображена на рис. 1. Здесь имеем 6-ти стадийный Pipeline на высшем уровне (компоненты Pipeline чаще упоминаются как “стадии”). Некоторые из этих стадий являются задачными компонентами (например, стадии 1 и 6), в то время как другие – иерархические скелетонные вложения (например, стадия 2 - Farm, в то время как стадия 3 - Deal).

Ниже показано представление технологической системы, основанное на скелетонах. Это описание является иерархическим — каждое поддерево описывается в глубину, слева направо поперек одного и того же уровня поддерева. К следующему поддереву переходим на том же уровне только после того, как все предыдущие поддеревья были описаны полностью, т.е. больше нет скелетонных вложений с неполными описаниями задач.

Представление технологической системы, изображенной на рис. 1 и 2, основанное на скелетонах, имеет вид:

```
pipe(6);
task("task", 1.0);
farm(3, "task", 3.0);
deal(2, "task", 2.0);
farm(3, "task", 3.0);
deal(3, "task", 3.0);
task("task", 1.0); .
```

Весь пример рассматриваем как пошаговую детализацию описания технологической системы, где мы представляем технологическую систему на высшем уровне и продолжаем уточнение до тех пор, пока описания самого низкого уровня не будут состоять только из задачных компонентов.

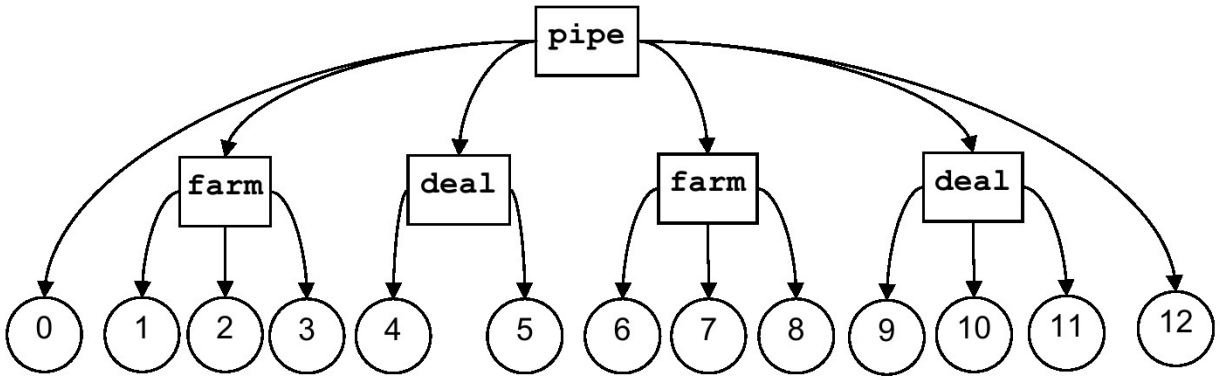


Рис. 2. Дерево иерархии скелетонов, которое соответствует примеру системы, показанной на рис. 1.

Для каждого структурированного приложения иерархическое дерево структуры данных внутренне поддерживается средством формирования модели. На эту структуру данных будем ссылаться как на *дерево иерархии скелетонов* (показанное на рис. 2 для примера технологической системы).

Дерево иерархии скелетонов включает в себя большую часть информации, представленной в описании (структурном описании в целом и при описании деталей компонентов технологической системы). Дополнительная информация, когда это необходимо, получается из этого дерева автоматически (например, граф зависимости данных, связанных с задачами).

3. Формирование моделей надёжности. Формирование моделей надёжности в терминах PEPA из заданного описания структурированного приложения может быть разделено на 3 этапа. На первом этапе, из дерева иерархии скелетонов получают ациклический ориентированный граф, который представляет зависимость данных между задачными компонентами. Этот граф используется на последующих этапах. На втором этапе формируются характеристики процесса для каждого элемента данных. Наконец, на третьем этапе, технологическая система в целом моделируется объединением задачных и скелетонных компо-

нентов, основываясь на их иерархической организации. Важность заключительного этапа состоит в том, что он дополняет модель надёжности установлением множеств синхронизации, которые будут использоваться пользователем модели в ходе синхронизации задачных компонентов на различных уровнях композиции.

Алгоритм 1 $GS(node)$: Формирование списков источников из дерева иерархии скелетонов.

$n := node.nchildren$ // Число узлов-потомков

$node.slist := parent.slist$ // Наследование списка источников от узла-родителя

$node.stype := parent.stype$ // Наследование типа схемы источника

for $i:=0$ **to** (n-1) **do**

$GS(node.child_i)$ // Рекурсивно формируемые списки источников для потомков

If $node.type$ is **task** **then** // Узел – задачный компонент

$v := \{x : where\ x = node.index\}$

if $parent.type$ is **deal** **or** **farm** **then** // Родитель - продублированный скелетон

$temp_{node} := v$

else

$parent.slist := v$

$parent.type := pipe$ // Обновление типа схемы источника

else if $node.type$ is **pipe** **then** // Узел – скелетон Pipeline

if $parent.type$ **deal** or **farm** **then** // Pipeline с дублированием

$temp_{node} := node.slist$

else

$parent.slist := node.slist$

$parent.stype := node.type$ // Обновление типа схемы источников

else if $node.type$ **deal** **or** **farm** **then** // Узел - продублированный скелетон

$m := \bigcup_{i=0}^{n-1} temp_i$ // Объединение временных списков по потомкам

if $parent.type$ **deal** **or** **farm** **then** // Дублирование с вложенным дублированием

$temp_{node} := m$

else

$parent.slist := m$

$parent.stype := node.stype$ // Обновление типа схемы источника.

3.1 Определение ациклического ориентированного графа. Пусть $G(\tau, \varepsilon)$ – ациклический ориентированный граф, где τ - множество задачных компонентов, а ε - множество ориентированных дуг, соединяющих задачные компоненты. Этот граф представляет зависимости данных, соответствующих дереву иерархии скелетонов. Для получения такого графа из заданного дерева иерархии скелетонов используются рекурсивные алгоритмы прямого обхода дерева.

Для каждой задачи в τ , введем индекс i, где $0 \le i < |\tau|$. Будем использовать обозначение t_i означающее: “задачный компонент с индексом i,” или, иногда, “задача i”. Конкретному графу G, связанному с каждой задачей t_i, ставят в соответствие два упорядоченных множества задач:

1.Список источников S_i , определяющий множество задач в τ , от которых каждая $i-$ая задача может получать данные;

2.Список назначений D_i, определяющий множество задач, которым каждая $i-$ая задача может отправлять данные. Эти упорядоченные множества определяются следующим образом:

$$S_i = \{j : (t_j, t_i) \in \varepsilon,\ i \neq j\}$$

и

$$D_i = \{j : (t_i, t_j) \in \varepsilon,\ i \neq j\}.$$

Если из контекста ясно, на какую задачу идёт ссылка, то нижний индекс в S_i и D_i опускаем. Важно отметить, что эти упорядоченные множества лишь перечисляют все возможные предшествующие, или последующие, задачи. Эффективное множество, с которым задача обменивается информацией, определяется из этих множеств путем применения соответствующего источникам, или назначениям, *способа взаимодействия*. Например, заданный Pipeline содержит два следующих друг за другом Deal. В общем случае все задачи в первом Deal будут в списке источников для второго Deal. Однако, если оба скелетона не одинакового размера, то фактически, задача во втором Deal будет получать данные только от той задачи в первом Deal, которая имеет такой же потомственный ранг внутри Deal. Поэтому эффективное множество - это *одноэлементное* множество.

Эти два множества в комбинации с задачным множеством τ, полностью характеризуют ациклический ориентированный граф G. Поэтому для формирования этих множеств из дерева иерархии скелетонов используют рекурсивные алгоритмы обхода дерева. В алгоритме 1 представлен процесс определения списков источников из дерева иерархии скелетонов. В этом алгоритме каждый узел сопровождается следующими параметрами:

$parent$ - указатель на родительский узел в

дереве иерархии скелетонов;

nchildren - количество узлов-потомков;

slist - список индексов задач, доступных этому узлу в качестве источников (фактически, это то, что будет определяться алгоритме);

stype - тип схемы источника;

type - тип узла, это может быть скелетонный узел или задачный узел (рис. 2).

Каждый узел также дополнительно сопровождается временным списком temp, который используется его узлом-родителем до полного формирования списка источников. Отметим, что для некоторых скелетонных узлов, где имеет место дублирование задач, список источников на этих узлах не может быть завершен до тех пор, пока все его поддеревья-потомки также не завершат формирование своих списков источников. Подобный алгоритм используется для формирования списков назначений.

3.2 Моделирование задачных компонентов. Задачный компонент в структурированных системах – это процесс, который многократно подвергается переходам: *получение→вычисление →передача*. В зависимости от структуры высшего уровня задачи, содержащей подзадачу, указываются соответственно эти три базовых действия. Иногда какие-либо из этих действий пропускаются. Например, в задачах-производителях, где не выполняется действие «получение»; или в задачах-потребителях, где не выполняется действие «передача».

В пункте 3.1 отмечено, что, когда одна задача взаимодействует с другими задачами, то перечень задач, участвующих во взаимодействиях, основан на эффективном подмножестве S (или D) и определяется характером взаимодействий. Этот характер взаимодействий основывается на скелетонном наследовании для данной задачи. Характер взаимодействия определяется как *функция* над списком источников. Эта функция выбирает задачные индексы из соответствующего списка, и, таким образом, формируется эффективное подмножество для текущего взаимодействия. Эта функция устанавливает порядок, при котором задача должна взаимодействовать с оставшимися задачами в дереве иерархии скелетонов. Функция источников установит, какие данные должны быть получены. Функция назначений определит, какие данные должны быть переданы.

Характер взаимодействия для данной задачи соответствует ее расположению в дереве иерархии скелетонов. В алгоритме 1 тип схемы источника $(stype)$ устанавливается по отношению к скелетонным компонентам, содержащимся в задаче. Тип схемы назначения устанавливается подобным же образом. Обозначив функции взаимодействия с источниками и назначениями, соответственно, как α и β, можно представить задачу как диаграмму переходов, изображающую последовательность обобщенных действий, выполняемых любой задачей в системе (рис. 3).

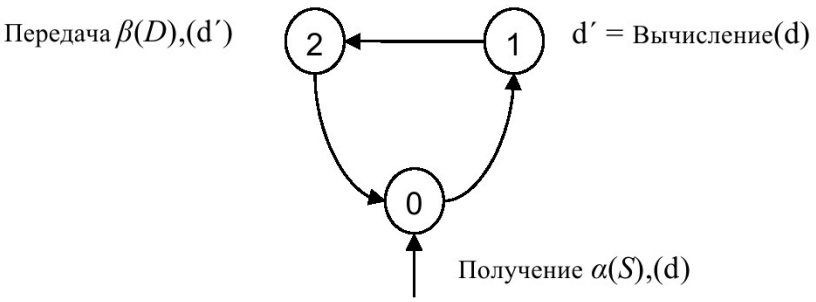


Рис. 3. Диаграмма переходов, которая дает последовательность обобщенных действий, выполняемых любой задачей в системе. В ходе формирования модели эти действия задаются, основываясь на характере взаимодействия задачи со своими предшественниками и преемниками.

Из этого абстрактного представления следует, что характеристика процесса задачных компонентов в терминах РЕРА определяется подмножествами $\alpha(S)$ и $\beta(D)$, а также, как это требуется согласно переходу «*получение→ вычисление→передача*», отношениями между α и β. Так как задачи могут иметь различные α и β, то необходимо установить схемы определений процессов для всех возможных комбинаций. Для краткости обозначим подобные комбинации как $\{\alpha(S) \to t_i \to \beta(D)\}$. Это означает, что "$i-$ая задача получает данные, основываясь на функции источников α, и передает данные, основываясь на функции назначений β". Когда какая-либо из функций не определена, то обозначим это действие *. Например, $\{* \to t_i \to \beta(D)\}$ для задачи-производителя.

После перечисления всевозможных ситуаций, сделаем на основе определений скелетонных конструкций несколько следующих кратких замечаний о взаимосвязях между различными функциями взаимодействий (См. пункт 2).

1. Функция взаимодействия *Deal* - это частный случай *Farm*, где вероятностный недетерминизм удален путем применения кругового распределения данных.

2. Функция взаимодействия *Pipeline* – это специальный случай *Deal*, где список источников является *одноэлементным* множеством.

Оставим без рассмотрения случаи, включающие *Pipeline*, так как их описания охватываются рассмотрением случаев с *Deal*. Рассмотрим комбинации функций взаимодействия *Deal* и *Farm*. **1.**

Ситуация $\{* \to t_i \to Deal(D)\}$. Здесь t_i - это задача-производитель. Она производит элементы данных, которые затем посылает одной из задач в D, выбираемых по правилу кругового распределения данных. Соответствующая характеристика процесса в терминах PEPA, где $n = |D|$ и $\lambda_i -$ вычислительное действие, сопоставляемое с t_i, выражается следующим образом:

$$t_i \overset{def}{=} (\lambda_i, \perp).(move_{i0}, \perp).(\lambda_i, \perp).(move_{i0}, \perp).\cdots$$
$$\cdots.(\lambda_i, \perp).(move_{i(n-1)}, \perp).t_1;$$

Здесь $move_{ij}$ представляет собой передачу данных от задачи i к j- ой задаче в D. Это обозначение выбираем вместо $send_{ij}$, так как эти действия будут использоваться далее, когда мы определим множества синхронизации. Когда принято последнее обозначение, необходимо определить систему для сопоставления соответствующих пар $send_{ij}$ и $receive_{ij}$..

2. Ситуация $\{Deal(S) \to t_i \to *\}$. В этом случае, t_i является задачей-потребителем. Она получает элементы данных от одной из задач в D, которые затем и использует. Следуя терминалогии, схожей с обозначением в предыдущем случае, имеем следующую характеристику процесса:

$$t_i \overset{def}{=} (move_{0i}, \perp).(\lambda_i, \perp).(move_{1i}, \perp).(\lambda_i, \perp).\cdots$$
$$\cdots(move_{(n-1)i}, \perp).(\lambda_i, \perp).t_1;.$$

Заметим, что $n = |S| -$ количество индексов задачи в списке источников. Для представления процедуры передачи данных от j- ой задачи в S к i- ой задаче используем $move_{ji}$.

3. Ситуация $\{Deal(S) \to t_i \to Deal(D)\}$.

Здесь t_i - задача-посредник. Полученные от одной из задач в S, элементы данных обрабатываются и результат передается одной из задач в D.В каждом случае коммуникаций по отправке и получению данных, эффективная задача выбирается автономно, основываясь на правиле кругового распределения данных.

В случае если количество элементов в списках источников и назначений одинаково, $p = |S| = |D|$, то имеем достаточно простую характеристику процесса:

$$t_i \overset{def}{=} (move_{0i}, \perp).(\lambda_i, \perp).(move_{i0}, \perp).\cdots$$
$$\cdots(move_{1i}, \perp).(\lambda_i, \perp).(move_{i1}, \perp).\cdots$$
$$\cdots(move_{(p-1)i}, \perp).(\lambda_i, \perp).(move_{i(p-1)}, \perp).t_1;.$$

В случае, если $|S| \neq |D|$, то не существует прямого соответствия между задачами-источниками и задачами-назначениями. Это несоответствие разрешается нахождением повторяющейся последовательности действий.

Определим периодичность p как число отдельных переходов «*получение→вычисление →передача*», после которых происходит повторение. Легко видеть, что периодичность – это *наименьшее общее кратное* $|S|$ и $|D|$. Имеем следующую характеристику процесса:

$$t_i \overset{def}{=} (move_{xi}, \perp).(\lambda_i, \perp).(move_{iy}, \perp).\cdots.$$

(повторить p раз, увеличивая k в каждой итерации). t_i;

где $0 \le k < p$, $x = k \bmod |S|$ и $y = k \bmod |D|$.

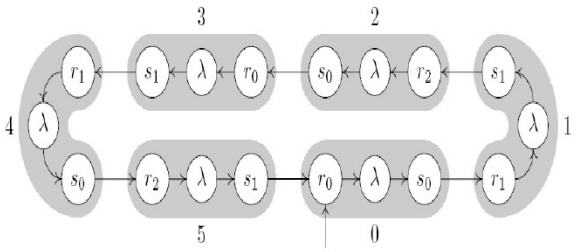


Рис.4. Диаграмма переходов, которая дает для каждого элемента данных, полученного по круговой диаграмме, обработанный результат. Он передается одной из задач, которая полностью завершила выполнение задания, назначенного ей ранее.

Например, когда $|S| = 3$ и $|D| = 2$, последовательность установившихся действий дает следующую характеристику процесса:

$$t_i \overset{def}{=} (move_{0i}, \perp).(\lambda_i, \perp).(move_{i0}, \perp).\cdots$$
$$(move_{0i}, \perp).(\lambda_i, \perp).(move_{i1}, \perp).(move_{2i}, \perp).\cdots$$
$$(\lambda_i, \perp).(move_{i0}, \perp).(move_{0i}, \perp).(\lambda_i, \perp).\cdots$$
$$(move_{i1}, \perp).(move_{1i}, \perp).(\lambda_i, \perp).(move_{i0}, \perp).\cdots$$
$$(move_{2i}, \perp).(\lambda_i, \perp).(move_{i1}, \perp).t_1;.$$

Три упомянутые ситуации можно использовать для полного определения задачных компонентов с любой комбинацией Pipeline и Deal.

Далее, представим ситуации, которые учитывают неопределенность, связанную со скелетоном Farm.

4. Ситуация $\{\Diamond \to t_i \to Farm(D)\}$. В этом случае задача является производителем. Однако, главное отличие здесь – это наличие неопределенности, которую фиксируем с помощью оператора *селекции* $(+)$ в терминах РЕРА. Это показывает следующая характеристика процесса:

$$t_i \overset{def}{=} (\lambda_i, \perp).t_i^1;$$

$$t_i^1 \overset{def}{=} (move_{i0}, \perp)t_i + (move_{i1}, \perp).t_i + \cdots$$

$$\cdots + (move_{i(n-1)}, \perp).t_1;$$

где $n = |D|$. После того, как элемент данных был получен, он передается любой из задач в D. Это возвращает задачу к состоянию воспроизводства данных.

5. Ситуация $\{Farm(S) \to t_i \to \Diamond\}$. В этом случае задача является потребителем. Характеристика процесса схожа с предыдущим случаем:

$$t_i \overset{def}{=} (move_{0i}, \perp)t_i^1 + (move_{1i}, \perp).t_i^1 + \cdots$$

$$\cdots + (move_{(n-1)i}, \perp).t_i^1;$$

$$t_i \overset{def}{=} (\lambda_i, \perp).t_i;$$

где $n = |S|$. После того, как элемент данных был получен от любой из задач в S, он используется. И задача возвращается в состояние получения данных.

6. Ситуация $\{Farm(S) \to t_i \to \{Farm(D)\}$.

В этом случае, t_i является задачей-посредником. Характеристика процесса для подобных задач может быть получена комбинированием двух предыдущих случаев:

$$t_i \overset{def}{=} (move_{0i}, \perp).(\lambda_i, \perp).t_i^1 + \cdots$$

$$\cdots (move_{(x-1)i}, \perp).(\lambda_i, \perp).t_i^1;$$

$$t_i \overset{def}{=} (move_{i0}, \perp).t_i + (move_{i1}, \perp).t_i + \cdots$$

$$\cdots + (move_{(y-1)i}, \perp).t_i;$$

где $x = |S|$ и $y = |D|$. Элементы данных принимаются от любой из задач в S, обрабатываются и результат передается любой из задач в D.

7. Ситуация $\{Deal(S) \to t_i \to Farm(D)\}$.

Здесь t_i является задачей-посредником. Особенностью этой задачи является то, что для каждого элемента данных, полученного по круговой диаграмме, обработанный результат передается одной из задач, которая полностью завершила выполнение задания, назначенного ей ранее. Здесь функцией источников является Deal. Задача i должна получать следующий элемент данных по круговой диаграмме. Задачи в S получают свои данные по круговой диаграмме . Как только данные были получены и обработаны, результат передается в Farm (рис. 4) . В ходе распределения используется композиция операторов селекции, что показано в следующей характеристике процесса:

$$t_i \overset{def}{=} (move_{0i}, \perp).(\lambda_i, \perp).t_i^0;$$

$$t_i^0 \overset{def}{=} (move_{i0}, \perp).t_i^1 + (move_{i1}, \perp).t_i^1 + \cdots$$

$$\cdots + (move_{i(y-1)}, \perp).t_i^1; \cdots$$

$$t_i^{x-1} \overset{def}{=} (move_{(x-1)i}, \perp).(\lambda_i, \perp).t_i^x;$$

$$t_i^x \overset{def}{=} (move_{i0}, \perp).t_i + (move_{i1}, \perp).t_i + \cdots$$

$$\cdots + (move_{i(y-1)}, \perp).t_i;$$

где $x = |S| - 1$ и $y = |D|$.

8. Ситуация $\{Farm(S) \to t_i \to Deal(D)\}$.

Этот случай подобен предыдущим, за исключением инверсии в расположении композиции селекции. Имеем следующую характеристику процесса:

$$t_i \overset{def}{=} (move_{i0}, \perp).t_i^0 + (move_{i1}, \perp).t_i^0 + \cdots$$

$$\cdots + (move_{(x-1)i}, \perp).t_i^0;$$

$$t_i^0 \overset{def}{=} (\lambda_i, \perp).(move_{i0}, \perp).t_i^1;$$

$$\cdots$$

$$t_i^{y-1} \overset{def}{=} (move_{0i}, \perp).t_i^y + (move_{1i}, \perp).t_i^y + \cdots$$

$$\cdots + (move_{(x-1)i}, \perp).t_i^y;$$

$$t_i^y \overset{def}{=} (\lambda_i, \perp).(move_{iy}, \perp).t_i;$$

где $x = |S|$ и $y = |D| - 1$.

Таблица 1

	*	Pipe	Deal	Farm
*	0	1	2	3
Pipe	4	5	6	7
Deal	8	9	10	11
Farm	12	13	14	15

В алгоритме 2 объединены все вышеописанные ситуации для формирования характеристик процесса для каждой из задач в дереве иерархии скелетонов. Для упрощенного представления каждой из этих ситуаций присвоен номер, как показано в табл.1. Здесь в столбцах перечисляются схемы назначений, а в строках - схемы источников.

В алгоритме 2 выражение $S(j)$ задает индекс m- ой задаче в S, где $m = j \bmod |S|$. Подоб-

ным образом определяется и соответствую-щее выражение для списка назначений $D(j)$. Для выделения частей сформированного определения процесса используется **Output**. При каждом инициировании **Output** все символы, следующие от него до конца строки, выделяют как часть характеристики процесса. Отметим значимость порядка инициации интерфейса **Output** для достоверности сформированной характеристики определения процесса.

Для формирования характеристик процессов всех задач в дереве иерархии скелетонов рассмотрим в нём все задачные узлы и для них используем алгоритм 2. Тем самым завершится второй этап формирования модели надёжности технологической системы. Перейдем к рассмотрению заключительного этапа, где определим множества синхронизации. В ходе определения этих множеств синхронизации будут использоваться действия $move_{ij}$ и $move_{ji}$, которые соответствуют передаче данных между задачами.

3.3 Моделирование системы. Все характеристики процессов, сформированные к концу второго этапа, моделируют только надёжность каждого задачного компонента независимо от других. А поскольку структурированное приложение – это совместное выражение этих задач, они должны быть синхронизированы соответственно уровню иерархической композиции. Это делается на заключительном этапе формирования модели.

На каждом уровне дерева иерархии скелетонов каждое поддерево соответствует закрытой подсистеме, где только пограничные задачные компоненты на каждой из сторон взаимодействуют со своими смежными родственными поддеревьями. Задачные компоненты, которые находятся внутри этой подсистемы, так называемые промежуточные компоненты, синхронизируются с другими задачными компонентами внутри этой же подсистемы. Здесь нет поперечной пограничной синхронизации. Заключительный этап формирова-ния модели происходит путем определения мно-жеств синхронизации между смежными поддере-вьями на каждом уровне дерева иерархии. Эти множества неоднократно уточняются до тех пор, пока все задачные компоненты не будут синхронизированы. [11-13].

Алгоритм 2 $GP(i)$: Формирование определения процесса для задачи i

Output: $t_i \overset{def}{=}$

1 :=**lcm** $(|S|,|D|)$ // Наименьшее общее кратное или сумма, если равно нулю

if case is 1 **or** 2 **then** // Предшественник - *, преемник - *Pipe* или *Deal*

for $0 \le j < 1$ **do Output**:

$(\lambda_i, \perp).(move_{i,D(j)}, \perp).$

else if case is 4 **or** 8 **then** // Предшественник - Pipe или Deal, преемник - *

for $0 \le j < 1$ **do Output**:

$(move_{S(j),i}, \perp).(\lambda_i, \perp).$

else if case is 5, 6, 9 **or** 10 **then** // Предшественник и преемник - Pipe или Deal

for $0 \le j < 1$ **do Output**:

$(move_{S(j),i}, \perp).(\lambda_i, \perp).(move_{i,D(j)}, \perp).$

else if case is 3 **then** // Предшественник - $\Diamond$, преемник - Farm

Output:

$(\lambda_i, \perp).t_i^1; \quad t_i^1 \overset{def}{=} (move_{i,D(0)}, \perp).t_i$

for $0 \le j < |D| - 1$ **do**

Output:

$+(move_{i,D(j)}, \perp).$

if $(|D| > 1) \Diamond (j < |D| - 1)$ **then Output**: t_i

else if case is 12 **then** // Предшественник - Farm, преемник - $\Diamond$

Output: $(move_{S(0),i}, \perp).t_i^1$

for $1 \le j < |S|$ **do**

Output: $+(move_{S(j),i}, \perp).t_i^1$

Output: ; $t_i^1 \overset{def}{=} (\lambda_i, \perp).$

else if case is 7 **or** 11 **then** // Предшественник - Pipe или Deal, преемник - Farm

for $0 \le j < |S|$ **do**

Output: $(move_{S(j),i}, \perp).(\lambda_i, \perp).t_i^j;$

$t_i^j \overset{def}{=} (move_{iD(0)}, \perp).;$

if $j < |S| - 1$ **then Output**: t_i^{j+1}

else Output: t_i

for $1 \le k < |D|$ **do**

Output: $+(move_{iD(k)}, \perp).$

if $j < |S| - 1$ **then Output**: t_i^{j+1}

else if $j < |D| - 1$ **then Output**: t_i

else if case is 13 **or** 14 **then** // Предшественник - Farm, преемник - Pipe или Deal

for $0 \le j < |D| - 1,$ начиная с $x = 0$ и увеличивая на 2 в каждом шаге **do**

Output: $(move_{S(0),i}, \perp).t_i^x$

for $1 \le k < |S|$ **do**

Output: $+(move_{S(k),i}, \perp).t_i^x$

Output: $t_i^x \overset{def}{=} (\lambda_i, \perp).(move_{iD(j)}, \perp).$

if $j < |D| - 1$ **then Output**: t_i^{x+1} ; $t_i^{x+1} \stackrel{def}{=}$

else if case is 15 **then** // И предшественник, и преемник - Farm

Output: $(move_{S(0),i}, \perp).(\lambda_i, \perp).t_i^1$

for $1 \le j < |S|$ **do**

Output: $+(move_{S(j),i}, \perp).(\lambda_i, \perp).t_i^1$

Output: $t_i^1 \stackrel{def}{=} (move_{iD(0)}, \perp).$

for $1 \le j < |D|$ **do**

Output: $t_i + (move_{iD(j)}, \perp).$

Output: t_i ;

Алгоритм 3 $GM(node, nchild)$: Формирование модели системы

$i := node.index$

$r := node.rank$ // Ранг узла между элементами одного уровня

if $node.type$ is **task then**

Output: t_i

if $r < nchild - 1$ **then**

if $parent.type \neq$ (**deal** or **farm**) **then**

Output $\rhd_L \lhd$, где

$L = \{move_{ij} : j = D_i(k), 0 \le k < |D_i|\}$

else

Output: ||

else

Output: (

for $0 \le j < node.nchild$ **do**

$GM(child_j, node.nchild)$ // Рекурсивно моделируемое поддерево

Output:)

if $r < nchild - 1$ **then**

if $parent.type \neq$ (**deal** or **farm**) **then**

Output $\rhd_L \lhd$ где

$L = \{move_{x,y} : y = D_i(j), x = S_y(k),$

$0 \le j < |D_i|, 0 \le k < |S_y|\}$

else

Output: ||

Для выполнения заключительного этапа применяют алгоритм 3. В этом алгоритме снова используется глубинный прямой обход дерева. Так как множество синхронизации между двумя поддеревьями может иметь отношение к одному из них, то выбирают ускоренный обход. Сначала определяется подсистема, соответствующая синхронизируемому поддереву, а затем синхронизируемое множество для поддерева. Каждый раз для задачного узла опускают эту задачу и формируют множества синхронизации, согласно которым эта задача синхронизируется со всеми ее задачами-преемниками. Когда задачный узел является скелетонным компонентом, то формируется множество синхронизации путем учета задач на границе отправки этой подсистемы. Они взаимодействуют с задачами на границе получения последующего поддерева.

4. Представление вычислительных и передающих информацию скоростей

Модель, которая сформирована к настоящему моменту, не завершена в двух аспектах. Скорости выполнения вычислений, а именно: вычислительные скорости, и скорости передачи информации, так называемые коммуникационные скорости, в явном виде не определены. Дополняют модель, представляя характерные активные оценки скоростей, так как в процессе выполнения синхронизации необходимы активные оценки скоростей передачи данных.

Так как метод формирования модели, предложенный здесь, предназначен преимущественно для распределенных приложений, последующие замечания будут сделаны в этом контексте. В распределенной системе, принципиальным фактором, определяющим скорость вычисления задачи, является скорость процессорного элемента, которому данная задача назначена для выполнения.

Отметим, что скорости передачи информации между задачами определяются через коммуникационные задержки лежащей в основе сети, связывающей эти процессорные элементы.

Для завершения модели имеют место два следующих момента. Основываясь на скорости процессорного элемента μ и скорости вычисления задачи λ, эту связь представляют следующим образом:

$$Ðrocessor_0 \stackrel{def}{=} (\lambda_0, \mu_0).Ðrocessor_0; \cdots$$

$$\cdots Ðrocessor_1 \stackrel{def}{=} (\lambda_1, \mu_1).Ðrocessor_1; \cdots .$$

Вычислительное действие λ_i, должно быть таким же, какое используется в определении задачи i. Подобным образом получают оценки скорости передачи информации, добавляя данные, которые берутся из матрицы смежности, представляющей собой коммуникационные задержки.

```
// Вычислительные скорости.
Processor_0 = (comp_0, 1.0).Processor_0;
Processor_1 = (comp_1, 3.0).Processor_1;
Processor_2 = (comp_2, 1.0).Processor_2;
Processor_3 = (comp_3, 2.0).Processor_3;
Processor_4 = (comp_4, 1.0).Processor_4;
// Передающие информацию скорости.
Network = (move_0_1, 1.0).Network + (move_1_2,
1.0).Network +
(move_2_3, 1.0).Network + (move_3_4,
1.0).Network;
// Определения задач.
```

```
t_0 = (comp_0, infty).t_01;
t_01 = (move_0_1, infty).t_0;
t_1 = (move_0_1, infty).(comp_1, infty).(move_1_2,
infty).t_1;
t_2 = (move_1_2, infty).(comp_2, infty).(move_2_3,
infty).t_2;
t_3 = (move_2_3, infty).(comp_3, infty).(move_3_4,
infty).t_3;
t_4 = (move_3_4, infty).(comp_4, infty).t_4;
// Модель системы.
Network<move_0_1, move_1_2, move_2_3,
move_3_4>(t_0 <move_0_1> t_1
<move_1_2> t_2 <move_2_3> t_3 <move_3_4>
t_4)<comp_0, comp_1, comp_2,
comp_3, comp_4>(Processor_0 || Processor_1 ||
Processor_2 ||
Processor_3 || Processor_4)
// Представление надёжности систе-
мы/пропускной способности
T1 = 1.0 * {** || (t_01 || t_1 || ** || ** || **) || (** || **
|| ** || ** || **)};
```

Рис. 5. Пример модели надёжности в терминах PEPA, сформированной автоматически из скелетонного представления.

Используемые действия $move_{ij}$ соответствуют характеристикам рассматриваемых задач (см. модель надёжности на рис. 5).

5. Выбор оборудования и схем углесосных станций

Анализ схем углесосных станций на действующих и проектируемых гидрошахтах указывает на их блочную структуру: в схеме предусматривается обычно столько абсолютно одинаковых и несвязных между собой блоков, сколько требуется при одновременной работе углесосов. Каждому работающему углесосу обычно придается два резервных. Связь основного и резервных углесосов с пульповодом осуществляется через систему задвижек и обратных клапанов.

Требуется определить коэффициент готовности и вероятность безотказной работы.

Используя изложенную выше методику структурного моделирования надежности технологических систем с применением скелетонов, основанную на полумарковских процессах, можно получить в замкнутой аналитической форме не только коэффициент готовности, но и другие характеристики блока.

На основе математических моделей надежности трех технологических схем углесосных станций произведём выбор лучшей из них (рис. 6-8).

Функционирование различных систем, рассматриваемых в теории надежности, может быть описано соответствующим образом построенным полумарковским процессом с конечным числом состояний [14].

Из определения полумарковского процесса следует, что при переходе из i- го в j- е состояние дальнейшее поведение процесса зависит от времени пребывания его в j-м состоянии. Это дает возможность составить систему стохастических уравнений, которая является аналогом формулы полной вероятности для случайных величин. Эти системы уравнений представляют основную математическую модель [15 - 17].

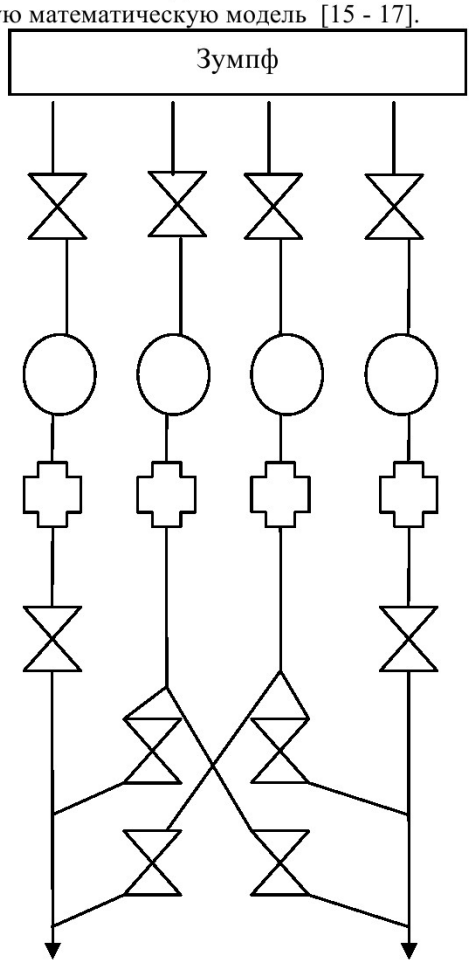


Схема I

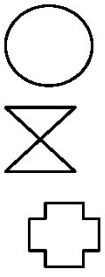
Рис. 6. Блок-схема 1 углесосной станции.

На основе полумарковских процессов могут быть получены различные характеристики надежности. Предполагаем, что каждый работающий элемент углесосной станции создает простейший поток отказов с параметром

$$\lambda = \frac{1}{T_p}$$

($\lambda_{àãä}, \lambda_{ê}, \lambda_{ç}$ – соответственно для агрегата, обратного клапана, задвижки).

На рис 6-8 приняты обозначения:



Углесосный агрегат

Задвижка

Обратный клапан

Время ремонта отказавшего элемента распределено по показательному закону с

$$\mu = \frac{1}{T_B}$$

($\mu_{агр}, \mu_к, \mu_з$ – соответственно для агрегата обратного клапана, задвижки).

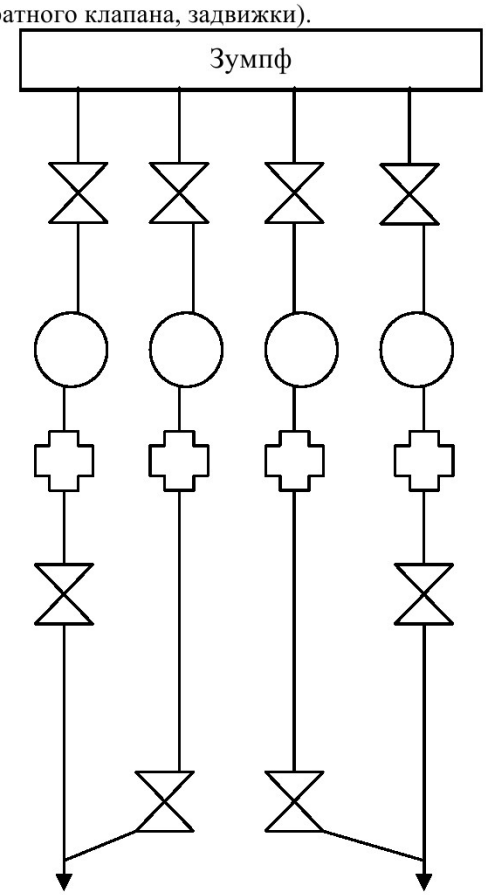


Схема II

Рис. 7. Блок-схема 2 углесосной станции.

Приведем математические модели надежности [16-19] для трех указанных схем как функции интенсивности отказов, интенсивности восстановлений, интенсивностей переключений технологической цепи.

Коэффициент готовности для схемы I :

$$\hat{E}_{\tilde{A}} = \frac{1}{1 + \dfrac{8\lambda_1^2\lambda_2}{(\mu+\mu_n)(\mu^2+2\mu\lambda_1+4\lambda_1^2)}}$$

Вероятность работы одной нитки трубопроводов для схемы I :

$$K^1 = \frac{1}{1 + \dfrac{(\mu-\lambda_1)\lambda_1^3\lambda_2}{(\mu^4-\lambda_1^4)(\mu+\mu_n)}}$$

Вероятность функционирования системы, приведенная по производительности технологической схемы I вычисляется по формуле :

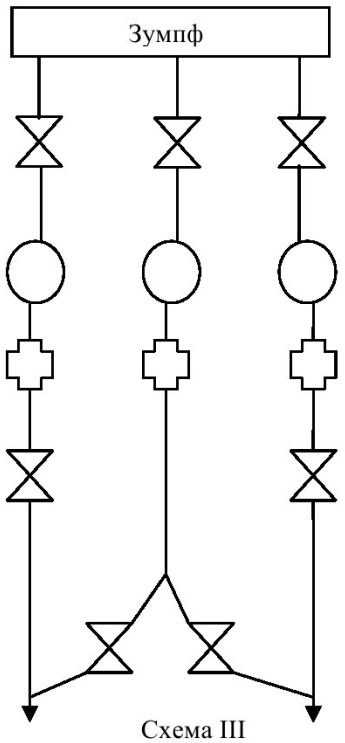


Схема III

Рис. 8. Блок-схема 3 углесосной станции.

$$P_{ф} = \frac{1}{2}\left(\frac{1}{1 + \dfrac{8\lambda_1^2\lambda_2}{(\mu+\mu_n)(\mu^2+2\mu\lambda_1+4\lambda_1^2)}} + \frac{1}{1 + \dfrac{(\mu-\lambda_1)\lambda_1^3\lambda_2}{(\mu^4-\lambda_1^4)(\mu+\mu_n)}} \right).$$

Наработка на отказ функционирования системы в случае схемы I :

$$T_0 = \frac{\mu^2 + 2\mu\lambda_1 + 4\lambda_1^2}{8\lambda_1^2\lambda_2}$$

Для автоматического формирования моделей надёжности заданная технологическая система, которая изображена на рис. 6. сначала должна быть представлена в форме, выражающей её сущность. Здесь применяется схемно-ориентированный подход, основанный на понятии алгоритмических скелетонов. Выше были перечислены алгоритмы и модели надёжности в терминах PEPA, сформированной автоматически из скелетонного представления.

Представим технологическую систему, которая изображена на рис. 6. Здесь имеем 8-тистадийный Pipeline на высшем уровне (компоненты Pipeline). Некоторые из этих стадий являются задачными компонентами (например, стадии 1 и 8), в то время как другие – иерархические скелетонные вложения (например, стадия 2 - Farm, в то время как стадии 3, 4 и 6 - Deal).

Покажем представление этой технологической системы, основанное на скелетонах. Это описание является иерархическим — каждое поддерево описывается в глубину, слева направо поперек одного и того же уровня поддерева. К следующему поддереву переходим на том же уровне только после того, как все предыдущие поддеревья были описаны полностью, т.е. больше нет скелетонных вложений с неполными описаниями задач.

Представление технологической системы, изображенной на рис. 6, основанное на скелетонах, имеет вид:

```
pipe(8);
task("task", 1.0);
farm(4, "task", 4.0);
deal(4, "task", 4.0);
deal(4, "task", 4.0);
farm(4, "task", 3.0);
deal(2, "task", 2.0);
deal(2, "task", 2.0);
task("task", 1.0); .
```

Весь пример рассматриваем как пошаговую детализацию описания технологической системы, где представляем технологическую систему на высшем уровне и продолжаем уточнение до тех пор, пока описания самого низкого уровня не будут состоять только из задачных компонентов.

Аналогично для схемы II :

Коэффициент готовности для схемы II :

$$K_{\Gamma} = \frac{1}{1 + \frac{(\mu - \lambda_1)\lambda_1^3\lambda_2}{(\mu + \mu_n)(\mu^4 - \lambda_1^4)}}$$

Вероятность работы одной нитки трубопроводов для схемы I I :

$$K^1 = \frac{1}{1 + \frac{\lambda(6\lambda^3 - \mu^3)}{\mu(\mu^3 + 3\lambda\mu^2 + 4\lambda^2\mu + 6\lambda^3)}}$$

Вероятность функционирования системы, приведенная по производительности технологической схемы I I вычисляется по формуле :

$$P_{\delta} = \frac{1}{2}\left(\frac{1}{1 + \frac{(\mu - \lambda_1)\lambda_1^3\lambda_2}{(\mu + \mu_n)(\mu^4 - \lambda_1^4)}} + \frac{1}{1 + \frac{\lambda(6\lambda^3 - \mu^3)}{\mu(\mu^3 + 3\lambda\mu^2 + 4\lambda^2\mu + 6\lambda^3)}} \right).$$

Наработка на отказ функционирования системы в случае схемы I I :

$$T_0 = \frac{\mu(\mu^2 + 2\mu\lambda + 2\lambda^2)}{2\lambda^2(\mu^2 + 3\mu\lambda + 3\lambda^2)}$$

Представим технологическую систему, которая изображена на рис. 7. Здесь имеем 6-тистадийный Pipeline на высшем уровне. Некоторые из этих стадий являются задачными компонентами (например, стадии 1 и 6), в то время как другие – иерархические скелетонные вложения (например, стадия 2 - Farm, в то время как стадии 3, 4 и 5 - Deal).

Представление технологической системы, изображенной на рис. 7, основанное на скелетонах, имеет вид:

```
pipe(6);
task("task", 1.0);
farm(4, "task", 4.0);
deal(4, "task", 4.0);
deal(4, "task", 4.0);
deal(4, "task", 4.0);
task("task", 1.0); .
```

Для схемы III имеем :

Коэффициент готовности для схемы I I I :

$$\hat{E}_{\hat{A}} = \frac{1}{1 + \frac{4\lambda_1\lambda_2}{(\mu + \mu_n)(\mu + 2\lambda_1)}}$$

Вероятность работы одной нитки трубопроводов для схемы I I I :

$$K^1 = \frac{1}{1 + \frac{\lambda_1^2\lambda_2}{(\mu + \mu_n)(\mu^2 + \mu\lambda_1 + \lambda_1^2)}}$$

Вероятность функционирования системы, приведенная по производительности технологической схемы I I I вычисляется по формуле :

$$P_{\delta} = \frac{1}{2}\left(\frac{1}{1 + \frac{4\lambda_1\lambda_2}{(\mu + \mu_n)(\mu + 2\lambda_1)}} + \frac{1}{1 + \frac{\lambda_1^2\lambda_2}{(\mu + \mu_n)(\mu^2 + \mu\lambda_1 + \lambda_1^2)}} \right).$$

Наработка на отказ функционирования системы в случае схемы I I I :

$$T_0 = \frac{\mu + 2\lambda_1}{4\lambda_1\lambda_2}$$

Представим технологическую систему, которая изображена на рис. 8. Здесь имеем 6-ти стадийный Pipeline на высшем уровне. Некоторые из этих стадий являются задачными компонентами (например, стадии 1 и 6), в то время как другие – иерархические скелетонные вложения (например, стадии 2 и 5 - Farm, в то время как стадии 3 и 4 - Deal).

Представление технологической системы, изображенной на рис. 8, основанное на скелетонах, имеет вид:

```
pipe(6);
task("task", 1.0);
farm(3, "task", 3.0);
deal(3, "task", 3.0);
deal(3, "task", 3.0);
farm(4, "task", 4.0);
task("task", 1.0); .
```

Получены для каждой схемы коэффициенты готовности $K_Г$ вероятности функционирования в системе только одной технологической цепи K' , вероятности функционирования, приведенные по производительности технологической схемы в целом $P_Ф$, наработки на отказ T_0 , вероятности безотказной работы схем как функции времени.

Среднее время работы технологической цепи определятся по формуле

$$T_{рц} = \frac{1}{(\lambda_{агр}+\lambda_{вс}+\lambda_k+\lambda_з)}. \qquad (1)$$

Среднее время восстановления технологической цепи определяется по формуле

$$T_{вц} = T_{рц}\left(\frac{\lambda_{агр}}{\mu_{агр}}+\frac{\lambda_{вс}}{\mu_{вс}}+\frac{\lambda_k}{\mu_k}\right). \qquad (2)$$

Так как полученные в данной работе математические модели характеристик надежности технологических схем являются многомерными моделями, то делают некоторые упрощения, которые позволяют представить их как функции одного безразмерного параметра:

$$\rho = \mu_ц / \lambda_ц \qquad (3)$$

где $\mu_ц$ -интенсивность восстановления (одной нитки) технологической цепи схемы; $\lambda_ц$ - интенсивность отказов технологической цепи.

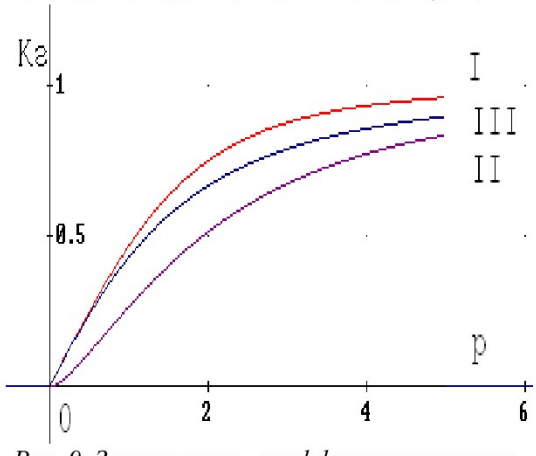


Рис. 9. Зависимость коэффициентов готовности $K_Г$ от удельного времени работы станций на единицу времени восстановления

Пренебрегая интенсивностями переключений $\mu_ц = \lambda_ц = 0$, получаем

$$\lambda_1 = \lambda_2 = \lambda_ц .. \qquad (4)$$

На рис. 9 представлены для каждой схемы коэффициенты готовности $K_Г$ как функции ρ. Из графика следует, что наиболее надежной схемой будет первая, а вторая и третья примерно равноценны для $\rho > 4$. Здесь же даны для каждой схемы вероятности функционирования, приведенные по производительности технологической схемы в целом $P_Ф$ как функции ρ. Наиболее высокой вероятностью функционирования обладает первая схема. Для $\rho > 3.5$ вероятности функционирования второй и третьей схем практически одинаковы.

Конкретный пример.

По данным лаборатории повышения надежности и износостойкости машин известно:

$$\lambda_{вс} = 0.00167; \quad \lambda_{агр} = 0.0238; \quad \lambda_k = 0.0021;$$

$$\lambda_з = 0.0038; \ \mu_{вс} = 0.333; \ \mu_{агр} = 0.843 ;$$

$$\mu_k = 1.43; \qquad \mu_з = 1.28. \qquad (5)$$

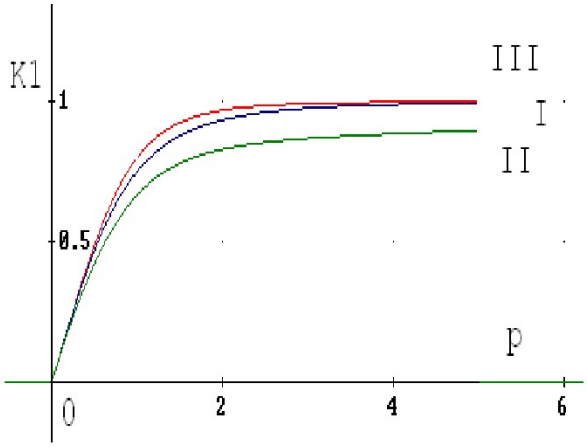


Рис. 10. Зависимость коэффициентов простоя $K_п$ от удельного времени работы станций на единицу времени восстановления

Отсюда с помощью формул (1–4) получают:

$T_{рц}$=29.3 часа; $T_{вц}$= 1.167; r = 25.1;

$K_{Гц}$=0.961.

Отсюда получаем значения коэффициентов готовности для каждой из схем.

Таблица 2

Схема	I	II	III
Кг	0,999535	0,996715	0,994152

Из табл. 2 видно, что для данных условий все схемы будут высоконадежными.

Для выбора наилучшей схемы надо учесть влияние надежности на экономичность технологическое схемы. Для компоновки схем I и II требуется примерно одинаковое число технологического оборудования, а для схемы III - на четверть меньше.

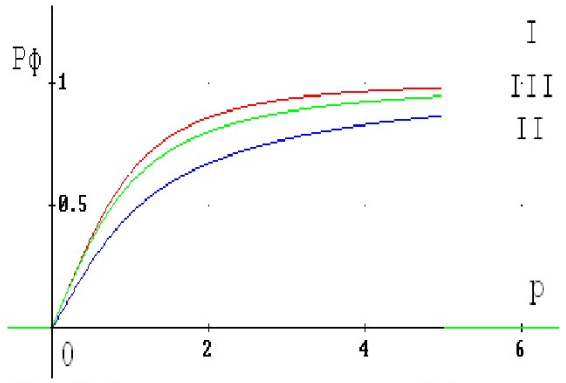


Рис.11. Зависимость вероятностей функционирования $P_Ф$ от удельного времени работы станций на единицу времени восстановления.

Итак, из указанных на рис. 9-11 технологических схем с учетом исходных данных (5), влияния надежности на экономичность и необходимого количества технологического оборудования в схеме устанавливаем оптимальность схемы III.

Заключение.

Количественный анализ полученных результатов показывает практические преимущества формирования моделей по сравнению с простыми системами, не обеспечивающими автоматического моделирования надёжности. Практическое применение подхода автоматического формирования модели состоит в динамическом планировании задач в параллельных и распределенных приложениях [18-19]. В данной работе рассмотрен автоматический подход для формирования РЕРА-модели надёжности из приложений, основанных на скелетонах. Подобные автоматические подходы важны в системах, где модель должна обновляться постоянно и динамически в зависимости от текущего состояния ресурсов. Показаны преимущества формирования моделей путем сравнения надёжности различных технологических схем .

СПИСОК ЛИТЕРАТУРЫ

1. *Aldinucci M, Danelutto M.* Algorithmic Skeletons Meeting Grids.// Parallel Computing, 32(7-8). 2006. p. 449–462.

2. *Rabhi F.A., Gorlatch S. editors*. Patterns and Skeletons for Parallel and Distributed Computing. // Springer Verlag, ISBN 1-85233-506-8, 2003.

3. *Benoit A. , Cole M,. Gilmore S. , Hillston J.* Flexible Skeletal Programming with eSkel.// In Euro-Par 2005, LNCS 3848, 2005. p. 761–770.

4. *Gomes C., Cunha J., Rana O.* Pattern Operators for Grid Environments.// Scientific Programming, 11. 2003. p. 237–261.

5. *Сорокин А. С.* Применение полумарковских процессов к определению характеристик надежности технологических схем. // Вестн. Кузбасского гос. тех. унив.,2005. № 1. С. 3 -9 .

6. *Сорокин А.С.* Применение методов теории вероятностей к исследованию некоторых процессов производства.//Труды 4-ой междунар. конф. Кибернетика и технологии XXI века. Воронеж, 2003. С. 312-323.

7. *Сорокин А.С.* Марковские процессы в теории надежности технологических систем гидродобычи угля // Вестн. Кузбасского гос. тех. унив., 2008. № 1. С.61-69.

8. *Cole M.* Algorithmic Skeletons: Structured Management of Parallel Computation.// MIT Press, Cambridge, Massachusetts, 1989.

9. *Hillston J.* A Compositional Approach to Performance Modelling.// Cambridge University Press, 1996.

10. *Cole M.* Bringing Skeletons Out of the Closet: A Pragmatic Manifesto for Skeletal Parallel Programming.// Parallel Computing, 30(3). March 2004. p. 389–406.

11. *Иглин С. П.* Grtheory toolbox — инструментарий matlab для решения задач на графах // Проектирование научных и инженерных приложений в среде MatLab. –М. : 2004. С. 165-185.

12. *Нечипуренко М. И., Попков В. К., Майнагашев С. М. и др.* Алгоритмы и программы решения задач на графах и сетях .// Новосибирск: Наука. Сиб. отд-ние, 1990. с. 515 .

13. *Пападимитриу Х., Стайглиц К.* Комбинаторная оптимизация. Алгоритмы и сложность. М.: Мир, 1985. с. 512 .

14. *Сорокин А.С.* Системы линейных уравнений. Основные понятия анализа. Полумарковские процессы. (Гриф УМО). Изд. СибГИУ. Новокузнецк, 1998. – 151 с.

15. *Сорокин А.С.* Оценка качества системы обслуживания технических объектов. //Качество высшего педагогического образования; уровни, параметры и критерии оценки. Сб. научных трудов КузГПА. Новокузнецк, 2004. с. 40 - 44.

16. *Сорокин А.С.* Алгоритм решения линейных систем дифференциальных уравнений Колмогорова. //Качество высшего педагогического образования; уровни, параметры и критерии оценки. Сб. научных трудов КузГПА. Новокузнецк, 2004. с. 45 - 48.

17. *Сорокин А.С.* Математическая модель: влияние надежности на экономичность// Инновации и информационные технологии в образовании. Труды Всероссийской конф. Т.2. Владивосток,

2004. с. 126 – 128

18. *Yaikhom G., Cole M., Gilmore S.* Combining Measurement and Stochastic Modelling to Enhance Scheduling Decisions for a Parallel Mean Value Analysis Algorithm. In V. Alexandrov, D. van Albada, P. Sloot, and J. Dongarra, editors, Intl. Conf. on Computational Science (ICCS 2006), volume 3992 of LNCS Springer, April 2006. p. 929–936.

19. *Benoit A., Cole M., Gilmore S., Hillston J.* Scheduling skeleton-based grid applications using PEPA and NWS.// The Computer Journal, 48(3). 2005. p. 369–378.

Автор статьи:

Сорокин
Андрей Семенович
- канд. физ.-мат.наук, доцент, ст.н.с.
(филиал КузГТУ , г. Новокузнецк)
Тел.8-3843-772459