N-ВЕРСИОННАЯ ПРОГРАММНАЯ АРХИТЕКТУРА С ДИНАМИЧЕСКОЙ СТРУКТУРОЙ

А. В. Новой

Сибирский государственный аэрокосмический университет

имени академика М. Ф. Решетнева, г. Красноярск, saneknov@yandex.ru

В статье рассматривается N-версионная программная архитектура и алгоритм голосования абсолютным большинством. Предлагается отказавшие версии на время восстановления исключать из программной архитектуры, а выбор решения осуществлять на основе результатов выданных оставшимися версиями. Такой режим работы позволяет повысить надежность программной системы.

В последнее десятилетие резко возросла роль программных средств (ПС) в жизни общества. Они применяются в системах управления и обработки информации и спектр решаемых ими задач весьма широк. Между тем с расширением сферы использования возросли требования к качеству программных систем. К критическим и сложным системам предъявляются самые высокие требования надежности, поэтому программные средства необходимые для их работы также должны отвечать этим требованиям.

Как известно любые сложные ПС содержат «скрытые» ошибки. Процесс тестирования не дает гарантии обнаружения всех имеющихся ошибок, поскольку количество возможных входных данных и их комбинаций велико, а проверить работу ПС на каждом наборе данных не представляется возможным. Даже наличие, на первый взгляд, незначительных ошибок может привести к катастрофическим последствиям. Так, например, в 1979 году американский космический зонд потерпел катастрофу из-за того, что в программе коррекции курса зонда запятая была спутана с двоеточием [1, с. 131].

Существуют различные методы проектирование программных средств, которые позволяют уменьшить влияние невыявленных ошибок на работу ПС. К таким методам относится метод проектирования отказоустойчивой архитектуры – N-версионное программирование.

N-версионное программирование предполагает разработку $N \geq 2$ версий компонента ПС, параллельное их исполнение, а затем сравнения результатов с помощью алгоритма голосования. Рассмотрим алгоритм голосования абсолютным большинством и покажем как, применяя его и изменяя структуру N-версионной программной архитектуры в процессе работы можно повысить надежность ПС.

Для принятия решения алгоритмом голосования абсолютным большинством необходимо чтобы не менее k:

$$k = ceil\left(\frac{N+1}{2}\right), \quad (1)$$

(ceil() – оператор округления до наибольшего целого числа) версий вернули эквивалентный результат [2, с. 15]. Другими словами на каждой итерации выполнения должно быть более половины одинаковых результатов возвращаемых версиями, иначе алгоритм голосования не сможет принять решение и ПС откажет. Рассмотрим работу алгоритма голосования на примере. Пусть N-версионная программная архитектура состоит из 4 версий, тогда для принятия решения необходимо 3 одинаковых результата, после отказа версии восстанавливаются. Предположим для упрощения ситуации, что одновременные отказы не происходят, и на каждой итерации отказывает по одной версии, получим:

итерация №1: все версии вернули одинаковый результат;

итерация №2: отказ первой версии, работоспособными остаются 3 версии;

итерация №3: отказ второй версии. В данном случае алгоритм голосования не сможет принять решения, поскольку работоспособными остались только 2 версии из 3 необходимых.

Представим процесс функционирования программной архитектуры в виде графа состояний (рис. 1), где l_i и n_i – интенсивность отказа и восстановления i версии. Каждое состояние обозначено вектором, в котором номер компоненты соответствует номеру версии, а значение компоненты характеризует работоспособность версии (1 – версия работоспособна, 0 – отказ версии).

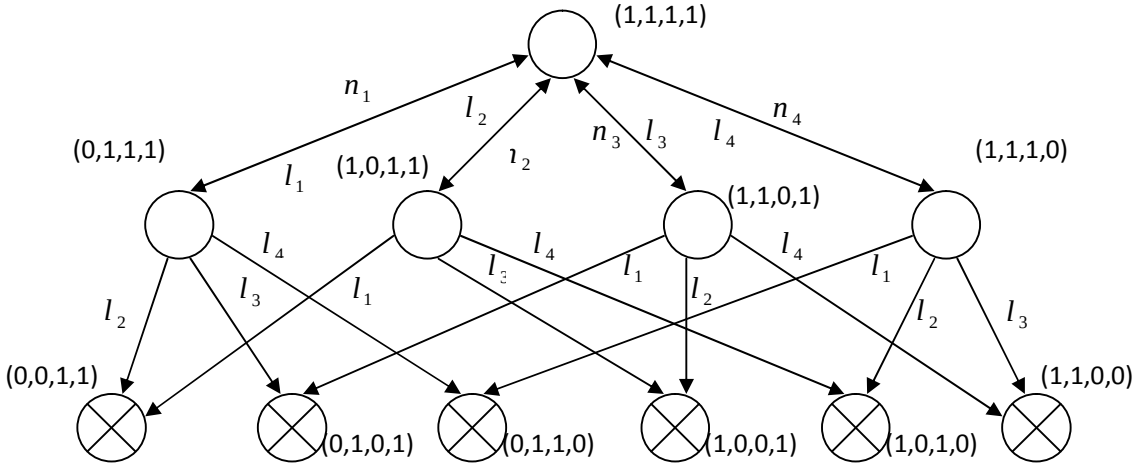


Рисунок 1 – Граф состояний программной архитектуры с четырьмя версиями

Граф состояний, представленный на рисунке 1, имеет 6 поглощающих состояний, в каждом из которых остается две работоспособные версии. Имеющиеся работоспособные версии необходимо использовать для продолжения функционирования программной системы. Предлагается исключать отказавшие версии из архитектуры ПС, тогда требуемое количество версий возвращающих одинаковый результат будет сокращаться, поскольку формула (1) остается прежней, а N уменьшается. По мере восстановления версии вводятся в архитектуру ПС снова. В таком режиме ПС может работать пока не останется две версии, и они не вернут разный результат. Покажем, как изменится процесс функционирования программной системы при динамическом изменении структуры программной архитектуры (рис. 2).

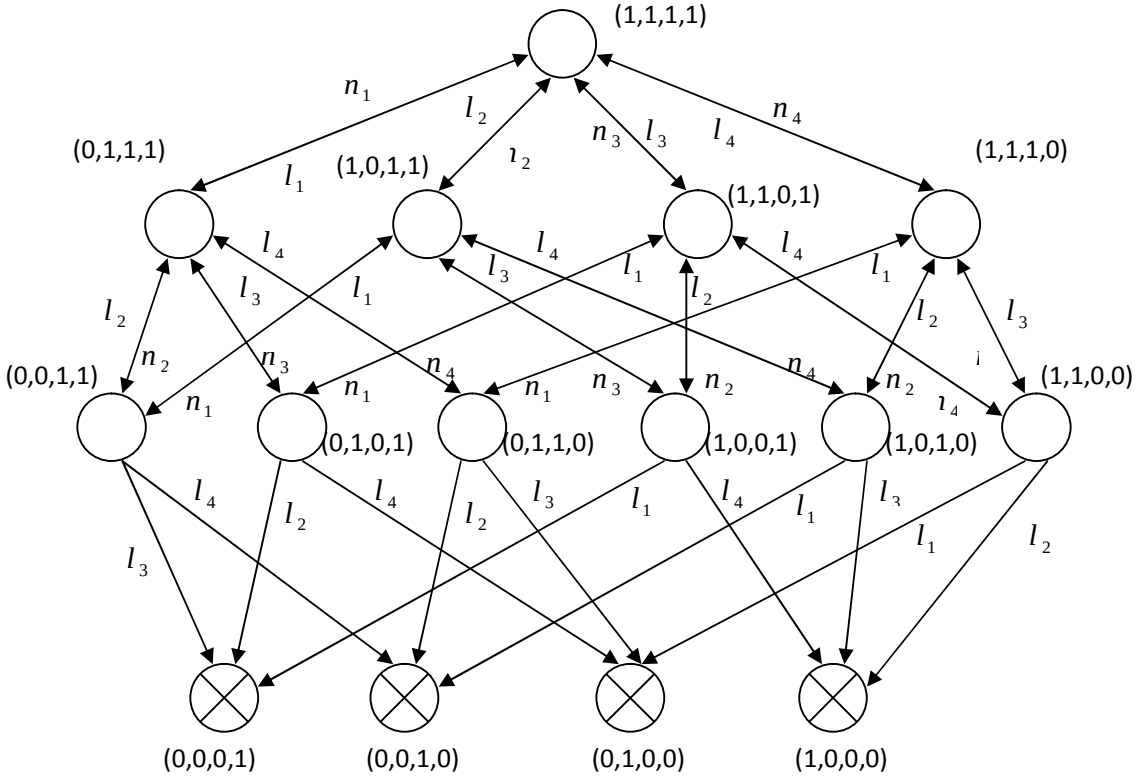
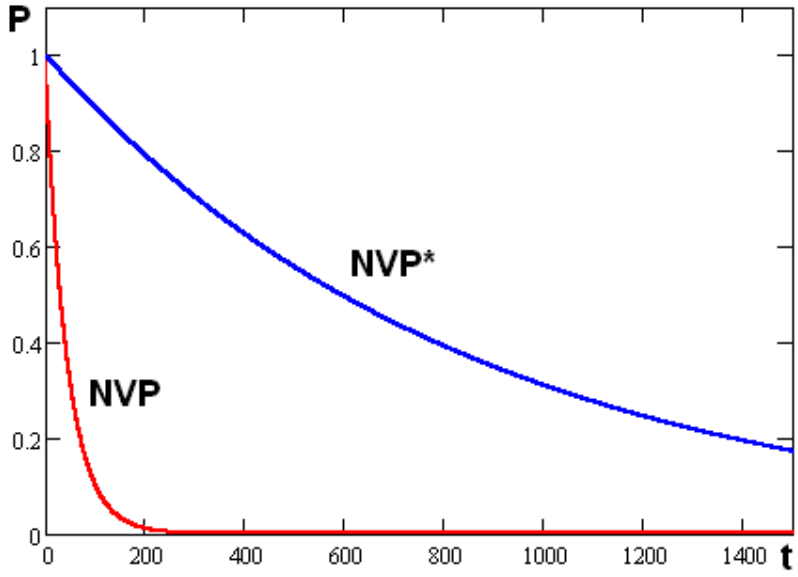


Рисунок 2 – Граф состояний программной архитектуры с динамической структурой

При использовании динамической структуры увеличивается количество работоспособных состояний, следовательно, увеличивается срок службы и повышается надежность ПС. Из графа, изображенного на рисунке 2, явно не прослеживается изменение структуры программной архитектуры. Это связано с тем, что каждое состояние по-прежнему определяется вектором с N компонентами, только увеличение количества состояний и возможных переходов демонстрируют структурные изменения в работе ПС.

Рассмотрим преимущества динамической структуры на примере. Пусть программная система состоит из четырех равнонадежных версий с параметрами: l = 0,05 и n = 1. На рисунке 3 представлены графики изменения вероятности безотказной работы ПС для двух архитектур.



NVP – обычная программная архитектура
NVP* - программная архитектура с динамической структурой

Рисунок 3 – Графики изменения вероятности безотказной работы

Из графиков (рис. 3) наглядно видно, что программная архитектура с динамической структурой обладает лучшими надежностными характеристиками.

Таким образом, динамическая структура *N*-версионной программной архитектуры позволяет, не меняя принципов работы алгоритма голосования, повысить надежность программной системы. Сравнительно простая модификация способа функционирования программной архитектуры дает значительный эффект, который наблюдается только при $N>3$.

СПИСОК ЛИТЕРАТУРЫ

1. Благодатских, В.А. Стандартизация разработки программных средств: учеб. пособие / В.А. Благодатских, В.А. Волнин, К.Ф. Поскакалов; под ред. О.С. Разумова. – М.: Финансы и статистика, 2006. – 288 с.

2. Новой, А. В. Расчет надежности отказоустойчивой архитектуры программного обеспечения / А. В. Новой, И. В. Ковалев // Вестник СибГАУ. Вып. 17. – 2007. – С. 14 - 17.