Автоматизация тестирования системы резервирования

Жердер В. М.
ОАО Московская биржа
Москва, Россия
Vadim.Zherder@moex.com

Ульянина Т. Ю.
НИЯУ МИФИ
Москва, Россия
Ulyanina@mail.ru

***Аннотация* — Рассмотрена задача автоматизации тестирования программного комплекса с двухуровневым резервированием на примере торговой системы биржи. Предложен подход на основе описания комплекса как системы конечных автоматов, тогда тестовые сценарии есть пути на графе переходов конечного автомата. На основе этого подхода создано инструментальное средство, позволяющее находить всевозможные пути графа (возможные сценарии осуществления переходов, порождающие управляющие bash-скрипты в операционной системе Linux). Предусмотрено исполнение порожденных скриптов в рамках инфраструктуры автоматизированного тестирования.**

Ключевые слова — системы конечных автоматов; резервирование; автоматизация тестирования; Python.

I. Введение

Задача обеспечения надежности нагруженных программно-технических комплексов остается актуальной в связи с ростом разнообразия архитектур подобных комплексов и развития используемых при их создании технологий. Существует обширная литература по проблемам повышения надежности систем в различных областях, таких как системы управления атомными электростанциями, летательными аппаратами и т.д. ([4]-[7]) Одним из основных методов повышения надежности является резервирование аппаратных и программных блоков.

Основными задачами в исследованиях, как правило, являются оптимизация конструктивных решений и оценка надежности построенных систем (см., например, [5][6][7]). Между тем, задача тестирования механизмов переключения на резервную систему сама по себе представляет интерес с точки зрения оценки времени переключения и правильности последовательности этапов переключения при различных сценариях возникновения нештатных ситуаций. Такое тестирование может осуществляться в рамках нагрузочного тестирования и предполагает моделирование ситуаций отказа основной системы. При условии небольшого количества возможных состояний элементов системы становится возможным обеспечить исчерпывающее тестирование (exhaustive testing) механизмов переключения, то есть рассмотрение всех возможных сценариев поведения системы в ситуации отказа. Важное значение имеет также возможность автоматизации тестов, особенно при использовании современных технологий разработки, таких как технологии непрерывной сборки (continuous integration, CI), которые предполагают автоматическую сборку новой версии программного обеспечения, ее развертывание на тестовый стенд и выполнение тестов.

В данной работе рассматривается торговая система Московской биржи, высоконагруженная система реального времени, обеспечивающая обработку биржевых транзакций. Поскольку биржа обеспечивает возможность совершения операций с финансовыми инструментами для большого количества участников рынка, включая центральный банк и правительство, то любые ошибки или остановки в работе торговой системы приводят к существенным финансовым потерям для участников торгов и самой биржи. Поэтому применяются различные методы повышения надежности торговой системы, в частности, двухуровневое структурное резервирование с использованием нагруженного («горячего») и полунагруженного («теплого») резервов [2].

Среди особенностей торговой системы можно отметить следующие:

- при отказе любого из составных модулей происходит аварийная остановка всей торговой системы, обработка транзакций прекращается. Поэтому можно рассматривать ее как единый функциональный модуль, соответственно, применяется общее резервирование.

- процессы сборки и развертывания торговой системы происходят быстро и не требуют существенных ресурсов, поэтому при разработке и модификации используется технология «непрерывной сборки», что, в свою очередь, требует наличие пакета автоматизированных тестов.

- при тестировании механизмов переключения на резервную систему мы можем не принимать во внимание особенности реализации системы и механизмов переключения, собственно функционал системы, допустимо тестирование системы как черного ящика.

Основными требованиями к торговой системе при тестировании надежности являются:

- непрерывность обработки транзакций;

- минимальные временные задержки при переключении с основной системы на резервную.

С этой точки зрения, мы можем рассматривать множество только тех состояний системы, которые характеризуют ее работоспособность и взаимодействия с резервными системами. Поэтому мы предлагаем описывать торговую систему как конечный автомат, тогда тестовые сценарии можно строить как пути на графе переходов. Такой подход позволяет реализовать алгоритм автоматического построения тестовых сценариев и обеспечить полное покрытие графа переходов при изменении набора состояний систем. Граф переходов также позволяет определять ожидаемые состояния системы после наступления тех или иных событий. Реальные последовательности переходов систем и временные задержки определяются по журналам работы после выполнения тестовых сценариев. Для анализа проведенного тестирования строится отчет – файл, содержащий таблицу с данными о переключениях компонент. Тест считается успешно пройденным, если последовательность переходов в новые состояния соответствовала ожидаемой, а временные задержки не превысили заданных максимальных величин. Таким образом, получен полностью автоматизированный комплекс, включающий инструменты создания тестового сценария, командный скрипт, реализующий этот сценарий, и построитель отчетов по журналам работы всех компонент.

Более детально задача сформулирована в части 2 данной статьи. В части 3 торговая система описана как конечный автомат, для него приведен граф переходов. В части 4 описана реализация автоматического создания и выполнения тестов с использованием языка Python и управляющих скриптов операционной системы. Пример отчета приведен в Приложении I.

II. Постановка задачи

В базовой конфигурации анализируемый комплекс состоит из нескольких компонент, размещенных на различных вычислительных узлах (обозначенных Tks1-Tks5), способных при наступлении определенных событий переходить в соответствующие состояния.

Начальная конфигурация системы из пяти компонент выглядит следующим образом (в квадратных скобках указано имя узла):

1. Главная компонента (Main) [Tks1].

2. «Горячий» резерв (backup, BU) [Tks2] – копия главной компоненты, постоянно поддерживает связь с главной компонентой, дублирует в реальном времени все транзакции. В случае недоступности главной компоненты происходит автоматическое переключение BU->Main. При этом все внешние подключения должны переключиться на новую главную систему.

3. «Теплый» резерв (warm backup, WBU) [Tks3] – копия главной компоненты, которая время от времени синхронизируется с Main и BU. С помощью специальной команды оператора может быть переведена в состояние BU.

4. Диспетчер (governor, Gov) [Tks4]– специальный процесс, который контролирует состояние Main, BU, WBU. Если какая-то из компонент теряет связь с другой, она запрашивает Gov, с тем, чтобы определить свое новое состояние.

5. Сервер доступа (gateway, GW) [Tks5] – транслирует запросы на выполнение транзакций от клиента в главную компоненту и результаты в обратном направлении. Клиент подключается к GW. При всех переключениях состояний Main, BU, WBU, сервер GW должен поддерживать соединение с главным на данный момент экземпляром главной компоненты.

Схема взаимодействий между компонентами приведена на Рис. 1.

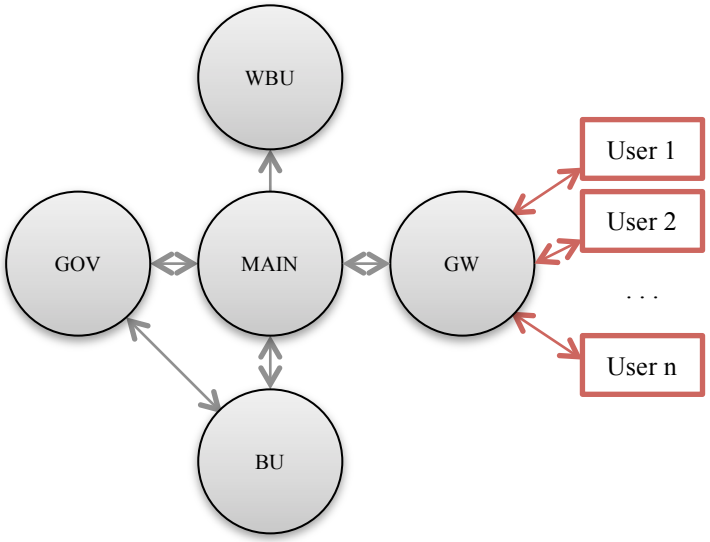

Рис. 1. Граф взаимодействий между компонентами.

В процессе тестирования реализуются всевозможные сценарии переключений компонент системы. После выполнения теста на каждом узле формируются журналы их работы.

Объем файлов таких журналов может быть велик (десятки мегабайт), что приводит к значительным временным затратам и трудностям при анализе журналов. В связи с этим, возникает необходимость автоматизации выполнения тестовых сценариев и анализа журналов в целях оптимизации тестирования.

Таким образом, требуется реализовать следующие задачи:

1) Создать сценарий теста – поток событий, внешних по отношению к системе.

2) Проверить адекватность реакции компонент на внешние события, то есть правильность последовательности переходов в новое состояние.

3) Убедиться в устойчивости подключения сервера доступа при всех изменениях состояния других компонент.

4) При изменении состояний компонент происходят временные задержки. Необходимо измерить величину временных задержек на сервере доступа при восстановлении подключения к новому экземпляру главной компоненты.

5) Измерить время переключения каждой компоненты в новое состояние и сформировать отчет о переходах компонент для всего теста.

Тест считается пройденным успешно, если GW не потерпел аварийного завершения и время восстановления подключения меньше заданного порогового значения (пороговое значение принимается равным 20 с).

III. Модель на основе конечных автоматов.

Каждая компонента анализируемого программного комплекса может принимать конечное количество состояний. Все возможные состояния компонент с пояснениями приведены в Таблице I. Переход из одного состояния в другое определяется событием, генерируемым в тестовом сценарии.

Для решения задачи предлагается рассматривать компоненты системы как конечные автоматы (event-driven finite state machine). [1][3] Так как нас интересует только переключение на резерв, то мы абстрагируемся от деталей функционирования, реализации взаимодействия и сводим число учитываемых состояний к минимуму. Полный список возможных состояний приведен в Таблице I.

ТАБЛИЦА I. Таблица возможных состояний компонент

№	Состояние	Описание
1	MAIN WITH BACKUP	MAIN работает с BACKUP, горячее резервирование в работе
2	BACKUP START	BACKUP в процессе синхронизации памяти
3	BACKUP READY	BACKUP в рабочем режиме, синхронизирован с MAIN
4	BACKUP WAIT GOVERNOR	BACKUP запрашивает разрешения продолжать работу в качестве MAIN
5	BU disconnected	Отсутствие разрешения на переключение BACKUP в режим работы MAIN, остановка обработки транзакций
6	MAIN SINGLE	MAIN работает без BACKUP
7	MAIN WAIT GOVERNOR	MAIN запрашивает разрешения продолжать работу (разрешение поступит, если BACKUP не переключался на MAIN)
8	WARMBACKUP START	Инициализация WARMBACKUP
9	WARMBACKUP READY	WARMBACKUP в рабочем режиме, синхронизирован с MAIN
10	WARMBACKUP CATCHUP MAIN	WARMBACKUP переключается в режим работы BACKUP
11	WARMBACKUP WAIT GOVERNOR	WARMBACKUP запрашивает разрешения продолжать работу в качестве BACKUP
12	WBU disconnected	Отсутствие разрешения на переключение WARMBACKUP в режим работы BACKUP, остановка обработки транзакций
13	Terminated	Аварийное завершение работы компоненты
14	Stop	Плановое завершение работы компоненты

Важно отметить, что все основные компоненты (MAIN, BU, WBU) являются копиями одного и того же конечного автомата, различающимися лишь начальными состояниями (см. Таблицу II).

ТАБЛИЦА II. Таблица соответствия компонент и начальных состояний на графе переходов

Компонента	**Начальное состояние**
MAIN	MAIN SINGLE
BU	BACKUP START
WBU	WARMBACKUP START

Полный граф переходов соответствует компоненте WBU, поэтому целесообразно рассмотреть лишь ее диаграмму состояний (см. Рис. 2). На графе также отмечены начальные состояния всех компонент.

Каждый сценарий теста – это путь на графе переходов. Таким образом, для осуществления тестирования необходимо произвести обход графа - построить всевозможные пути самого разветвленного графа, компоненты WBU, по которым будут реализовываться тестовые сценарии.

В Таблице III приведен пример сценария теста, соответствующий ему путь на графе на Рис. 2 отмечен выделенными дугами.

Для упрощения представления схемы множество конечных состояний обозначается на диаграмме одним объектом, разделенным на две части. Это означает, что система приходит в одно из этих состояний в зависимости от внешних сигналов: состояние Terminated достигается аварийным завершением работы, Stop – плановым остановом работы компоненты.

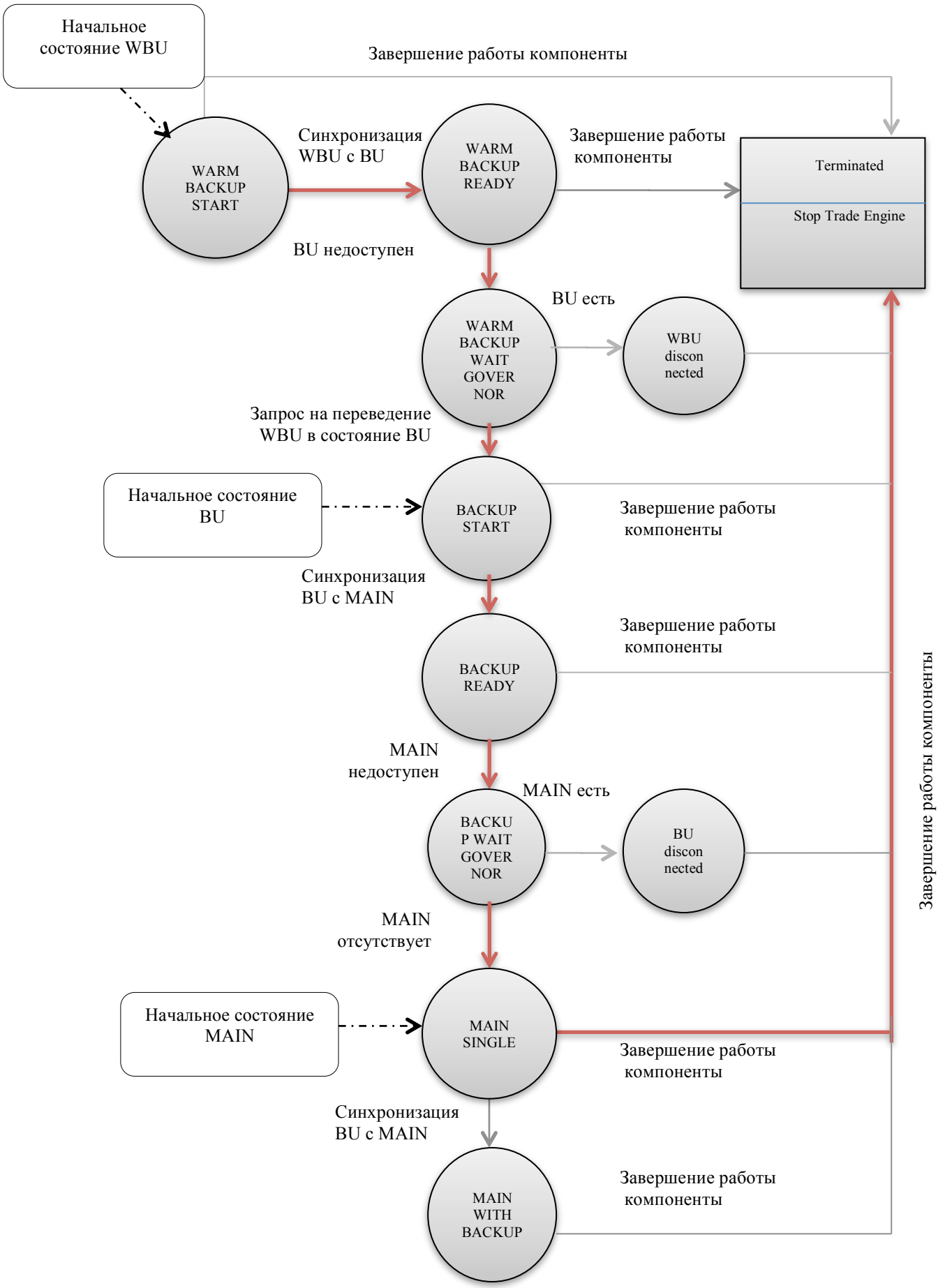

Рис. 2. Граф переходов компоненты WBU

ТАБЛИЦА III. ПРИМЕР СЦЕНАРИЯ, ПОЛУЧЕННОГО ПО ГРАФУ ПЕРЕХОДОВ

Event	MAIN state	BU state	WBU state
	MAIN SINGLE	BACKUP START	WARMBACKUP START
Sync BU with MAIN	MAIN WITH BACKUP	BACKUP READY	WARMBACKUP START
Sync WBU with BU	MAIN WITH BACKUP	BACKUP READY	WARMBACKUP READY
BU unaccessible	MAIN SINGLE	No BU	WARMBACKUP WAIT GOVERNOR
Gov Permission granted	MAIN SINGLE	No BU	BACKUP START
Sync BU with MAIN	MAIN WITH BACKUP	No BU	BACKUP READY
MAIN unaccessible	No MAIN	No BU	BACKUP WAIT GOVERNOR
Gov Permission granted	No MAIN	No BU	MAIN SINGLE

IV. РЕАЛИЗАЦИЯ СИСТЕМЫ АВТОМАТИЗИРОВАННОГО ТЕСТИРОВАНИЯ РЕЗЕРВИРОВАНИЯ НА ЯЗЫКЕ PYTHON

Модель компоненты программного комплекса реализована на языке Python с помощью специального модуля FySom [8], предназначенного для работы с конечными автоматами. Модуль позволяет хранить граф переходов в виде словаря событий, в котором хранятся сами события и соответствующие им исходные и конечные состояния. Созданный на основе модуля FySom класс Component предусматривает дополнительно хранение имени компоненты, журнала ее работы и параметров узла, на котором она запускается. Это позволяет создавать экземпляры класса, соответствующие компонентам системы, и контролировать их состояния независимо друг от друга.

1. Создание сценария

Сценарий теста – путь на графе переходов. Для поиска всевозможных путей графа, по которым будут проигрываться тестовые сценарии, необходим обход графа. Обход графа осуществляется с помощью рекурсивной функции, которой в качестве аргумента передается начальное состояние компоненты. Функция осуществляет поиск всевозможных путей обхода графа для определенной компоненты, полученный сценарий как последовательность событий записывается в файл. В файл также записывается последовательность переходов компонент по состояниям: каждая колонка содержит пройденные своей компонентой состояния, каждое состояние соответствует событию в соответствующей строке, в результате которого она была достигнута. В Таблице III приведена выдержка из файла со сценариями – один из сценариев.

Сценарии тестов порождают управляющие bash-скрипты в операционной системе Linux.

Предусмотрена возможность выбора тестового сценария с помощью ввода соответствующего сценария.

2. Выполнение сценария

Порожденные скрипты исполняются в рамках инфраструктуры автоматизированного тестирования.

В процессе работы скрипт контролирует состояния компонент. В случае, если ситуация не соответствует ожиданиям, скрипт прекращает работу и генерирует сообщение об ошибке, таким образом проверяется адекватность реакции компонент на внешние события. Ожидаемые состояния определяются по графу переходов и записываются вместе с последовательностью событий в файл со сценариями.

3. Анализ результатов теста

Вычисление величины временных задержек на сервере доступа при восстановлении подключения к новому экземпляру главной компоненты и анализ результатов переключений всех компонент производится по ключевым словам в файлах журналов их работы [1]. Полученные данные о переходах компонент заносятся в файл отчета (см. Приложение I). Файл отчета содержит таблицу, в которую последовательно заносится каждое событие и имя узла, на котором произошел описываемый переход компоненты, начальное состояние компоненты, время начала перехода, время завершения, конечное состояние и промежуток времени, за который данный переход произошел.

Дополнительно по журналу работы сервера доступа GW проверятся непрерывность подключения: время разрыва соединения не должно превышать заданного порогового значения.

[1] Ключевые слова, использованные в работе, немного изменены для наглядности.

V. Заключение

В работе рассмотрен программный комплекс торговой системы Московской биржи как высоконагруженная система с двухуровневым резервированием. Для реализации задачи автоматизации тестирования механизма переключения на резерв система описана как конечный автомат, а тестовые сценарии представлены как пути на графе переходов. Разработаны инструменты, реализующие следующие возможности:

- генерацию всевозможных тестовых сценариев путем обхода графа переходов конечного автомата;

- анализ правильности реакции компонент на внешние события;

- анализ устойчивости подключения сервера доступа при всех изменениях состояния других компонент;

- вычисление времени переключения каждой компоненты в новое состояние;

- формирование отчета о переходах компонент для каждого теста.

Результаты работы внедрены в практику тестирования на Московской бирже.

Литература

[1] Wagner F., Schmuki R., Wagner Th., Wolstenholme P. Modeling software with finite state maschines. A practical approach., Auerbach Publications Taylor & Francis Group, Boca Raton, 2006

[2] Черкасов Г.Н. Надежность аппаратно программных комплексов: учебное пособие /Т.Н. Черкасов - СПБ.: Питер 2005

[3] Карпов Ю.Г. Теория конечных автоматов Питер 2003

[4] Половко А.М., Гуров С.В. Основы теории надежности. Санкт-Петербург. "БХВ Петербург", 2006

[5] Р. В. Нестуля, О. В. Сердюков, А. Н. Скворцов. Архитектура отказоустойчивой распределенной среды управления для АСУТП крупных технологических объектов. Труды VI Международной конференции «Параллельные вычисления и задачи управления» (PACO’2012)

[6] Максименко С.Л., Мелехин В.Ф. Анализ надежности цифровых устройств со структурным резервированием и периодическим восстановлением работоспособного состояния узлов. Информационно-управляющие системы Выпуск№ 3 (64) 2013

[7] Пьо Маунг Ко. Оптимизация безотказности систем управления летательных аппаратов при активном нагруженном резервировании. Диссертация на соискание ученой степени кандидата технических наук. М.: ГТУ МАИ, 2009

[8] Модуль fysom для работы с конечными автоматами. http://wiki.python.org/moin/FiniteStateMachine

Приложения

ПРИЛОЖЕНИЕ I. Выдержки из отчета о переключениях компонент

Event / TKS node	Initial State	Time 1	Time 2	New State	Time delta
Sync BU with MAIN /Tks1/	MAIN SINGLE	2014-06-14 10:16:21.786970	2014-06-14 10:16:37.254394	MAIN WITH BACKUP	15.467424
Sync BU with MAIN / Tks2/	BACKUP START	2014-06-14 10:16:34.228663	2014-06-14 10:16:37.254197	BACKUP READY	3.025534
Sync WBU with BU / Tks3/	WARMBACKUP START	2014-06-14 10:17:01.699093	2014-06-14 10:17:01.699386	WARM BACKUP READY	0.000293
MAIN unaccessible /Tks2/	Lost link to Main	2014-06-14 10:23:00.330469	2014-06-14 10:23:02.001329	MAIN SINGLE	1.67086
Sync BU with MAIN /Tks2/	MAIN SINGLE	2014-06-14 10:23:02.001329	2014-06-14 10:23:06.374270	MAIN WITH BACKUP	4.372941
WBU to BU request /Tks3/	WARMBACKUP CATCHUP MAIN	2014-06-14 10:23:03.345932	2014-06-14 10:23:03.345965	BACKUP START	0.00033
Sync BU with MAIN /Tks3/	BACKUP START	2014-06-14 10:23:03.345965	2014-06-14 10:23:06.373897	BACKUP READY	3.027932