УДК 004.415.5

И.Б.Туркин, Б.Б. Михнич

ФОРМАЛЬНАЯ ВЕРИФИКАЦИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ АВТОМАТИЗАЦИИ ИСПЫТАНИЙ СИСТЕМ КОСМИЧЕСКИХ АППАРАТОВ

Национальный аэрокосмический университет им. Н.Е. Жуковского “ХАИ”
energy@d4.khai.edu

Рассматриваются особенности применения технологии Windows Workflow Foundation при разработке программного обеспечения для автоматизации стендовых испытаний подсистем космических аппаратов. Показано, что применение данной технологии позволяет технологу непосредственно описать ход вычислительного процесса, используя базовые исполняемые блоки или шаблоны в виде потоков работ, что повышает информативность и упрощает дальнейшие модификации системы. Рассматривается механизм верификации приложений, построенных по спецификации Windows Workflow Foundation и способных изменять бизнес-логику на этапе выполнения. Предложен механизм преобразования процесса в нотации сетей Петри. Для верификации предлагается использовать математический аппарат сетей Петри, анализируя свойства бездефектности замкнутой эквивалентной сети Петри. Дано определение требований, характеризующих бездефектную сеть Петри.

Розглядаються особливості застосування технології Windows Workflow Foundation при розробці програмного забезпечення для автоматизації стендових випробувань підсистем космічних апаратів. Показано, що застосування даної технології дозволяє технологу безпосередньо описати хід обчислювального процесу, використовуючи базові блоки або шаблони у вигляді потоків робіт, що підвищує інформативність і спрощує подальші модифікації системи. Розглядається механізм верифікації програм, побудованих за специфікацією Windows Workflow Foundation і здатних змінювати бізнес-логіку під час виконання. Запропоновано механізм перетворення робочих потоків у нотацію мереж Петрі і наведений приклад такого перетворення. Для верифікації запропоновано використовувати математичний апарат мереж Петрі, шляхом аналізу властивостей бездефектної замкнутої еквівалентної мережі Петрі. Дано визначення вимог, що характеризують бездефектну мережу Петрі.

Features of application of Windows Workflow Foundation technology on software development for test bench automatization of such spacecraft subsystem. There was shown that application of given technology allows the technologist to describe directly computational process flow, using base run-units (executed units) or templates in the form of working streams. This increases clearness and simplifies further system modification. The mechanism of verification of working threads of the applications constructed under specification Windows Workflow Foundation and capable to change business logic at a performance stage is considered. The mechanism of transformation of working threads in the notation of Petri nets offered and the example of such transformation is resulted. For verification it is offered to use a mathematical apparatus of Petri nets, analyzing properties faultlessness closed equivalent Petri net. Definition of the requirements characterizing faultlessness Petri net is made.

Ключевые слова: космический аппарат, автоматизация испытаний, программное обеспечение, формальные методы верификации, модель потоков работ, сеть Петри

Введение

За годы независимости в Украине были успешно реализованы три космические программы, а в настоящее время реализуется четвёртая общегосударственная космическая программа Украины на 2008-2012 года, которая имеет статус Закона Украины [1]. Выполнение программы даст возможность создать современные космические системы, в том числе «Сич-2М», для наблюдения за Землёй и геофизического мониторинга, спутниковые телекоммуникации сетей связи и вещания общего пользования, систему геоинформационной связи (как часть европейской системы GMES и мировой GEOSS), разработку космических аппаратов нового поколения и т.д. Существенной составной успешной работы по этим проектам есть создание высокоэффективных и надёжных космических аппаратов (КА).

Остается актуальной научно-техническая проблема создания современных и конкурентоспособных систем КА в условиях быстро развивающихся запросов рынка космической отрасли, когда высокое качество конечного продукта должно быть достигнуто за минимальное время с ограниченными ресурсами. Решение данной проблемы требует комплексного развития материаловедения и технологии производства, численных методов

анализа и поиска оптимальных проектных решений, теории и практики испытаний, развития производственной и стендовой базы, информатизации всех этапов жизненного цикла. Все этапы жизненного цикла производства КА подлежат автоматизации, и, как следствие, необходимы высокотехнологичные программные комплексы.

Автоматизация стендовых испытаний ракетно-космической техники – одно из наиболее важных средств повышения ее надежности, наряду с резервированием, технической диагностикой и аварийной защитой. Одна из главных особенностей программного обеспечения (ПО) для автоматизации стендовых испытаний КА заключается в необходимости частой модификации алгоритмов и типов проверок аппаратуры [2].

Создание корректных и безотказных систем невозможно без применения методов формальной верификации программного обеспечения, под которой будем понимать формальное доказательство на абстрактной математической модели корректности алгоритмов управления, при этом предполагается, что соответствие между математической моделью и структурой системы считается изначально заданным. В современной программной инженерии появляются новые подходы к разработке программных продуктов, которые активно используются разработчиками программного обеспечения. Один из таких подходов основан на технологии Windows Workflow Foundation (WWF), которая ориентирована на визуальное проектирование и использует декларативную модель программирования. Для такого подхода становится необходимым и возможным верификация каркасной логики разрабатываемого приложения.

1. Анализ исследований и публикаций

Малые беспилотные КА (микроспутники), конструируемые КБ «Южным», включают в себя 20...25 систем различных наименований и назначений: системы управления, телеметрии, ориентации, терморегулирования, антенно-фидерная и др. Каждая из этих систем, в свою очередь, имеет достаточно сложную структуру и состоит из подсистем, приборов, агрегатов, датчиков. Подсистемы соединены между собой и осуществляют информационный обмен посредством подсистемы данных платформы (ПДП). Сложность испытаний, как этапа жизненного цикла КА, объясняется тем, что в это время не только контролируется правильность сборки изделия, проверяется исправность и логика функционирования его систем и компонентов, но и одновременно вводится в эксплуатацию контрольно-измерительная и контрольно-проверочная аппаратура, проверяется эксплуатационная документация, создается собственно технологический процесс испытаний [3].

Состояние современного КА определяется тысячами параметров, характеризующих тепловые, пневматические, гидравлические, химические, электрические явления, и сотни - тысячи команд требуется для управления. Интенсивность информационных потоков при испытаниях может достигать несколько мегабайт в секунду. Динамика переходных процессов различных систем КА колеблется от долей секунд до суток. В некоторых случаях скорость принятия решений по управлению изделием должна быть практически мгновенной. В случае отклонения параметров от нормы формируются конкретные рекомендации по управлению тестируемой подсистемы и КА для обеспечения выполнения КА целевой задачи.

Примерами математических объектов, часто используемых для моделирования систем [4], являются: системы переходов; конечный автомат; помеченная модель состояний и переходов; сеть Петри [5]; временной автомат [6]; гибридный автомат; алгебра процессов; формальная семантика языков программирования, например операционная семантика, денотационная семантика, аксиоматическая семантика и логика Хоара; темпоральная логика [7].

В настоящее время появляется много новых технологий разработки программного обеспечения, нацеленных на то, чтоб процесс разработки стал менее трудозатратен, а конечный продукт наиболее надёжен и прост в сопровождении. Следует выделить на фоне всего многообразия технологию Windows Workflow Foundation (WWF), которая является ключевым компонентом платформы Microsoft .NET Framework, начиная с версии 3.0, поставляемой в составе Visual Studio [8].

WWF – одна из фундаментальных технологий компании Microsoft, входит в состав .NET Framework 3.0 [1], который

изначально установлен в Windows Vista и может быть установлен в Windows 2003 Server и Windows XP SP2. Данная технология позволяет определять, запускать на выполнение и управлять рабочими процессами. WWF предоставляет объектную модель и средства разработки приложений, основанных на модели потоков работ, и может использоваться в самом широком спектре сценариев – от взаимодействия с пользователем до управления распределенными бизнес-процессами, обеспечивает визуальное и декларативное построение потоков операций на основе управляемого кода. При разработке приложений с использованием технологии WWF профессиональным программистом в среде .Net имеются встроенные средства верификации потоков работ. Процессы Workflow можно определить двумя способами: императивно, т.е. на любом из .NET-языков, например, C# или VB.NET; декларативно на XAML (eXtensible Application Markup Language) – язык интерфейсов. В WWF при помощи XAML можно определять последовательности выполняемых действий (workflows), редактирование которых возможно как в графическом, так и в текстовом виде.

WWF предоставляет возможность модификации потоков работ конечным пользователем в виде дополнительного инструментария настройки бизнес логики приложения. Инструментарий конечного пользователя является дополнительным средством адаптирования приложения и обычно не содержит развитых встроенных средств контроля потоков работ, поэтому предлагается организовать анализ корректности описанных потоков работ, особенно в случае автоматического формирования технологического процесса испытаний.

Поскольку в WWF процессы описываются в виде графических схем, возможен их перевод в нотацию сетей Петри и последующий качественный анализ с применением математического аппарата. Язык сетей Петри обладает формальной семантикой, наглядным графическим представлением, выразительностью. Его развитый математический аппарат позволяет проверять множество свойств моделируемых потоков работ и вырабатывать разнообразные методы для их анализа.

2. Постановка задачи

Целью этой статьи является анализ возможности использования технологии WWF в задачах разработки ПО для автоматизации стендовых испытаний КА и обоснование возможности применения сетей Петри для верификации моделей WWF. Предлагается анализ возможности применения четкого математического аппарата для верификации разрабатываемых процессов WWF, который позволяет качественно оценить рассматриваемый алгоритм и вовремя выявить ошибки.

3. Результаты исследований.

3.1 Анализ принципов реализации модели потоков работ в среде WWF.

Архитектура WWF состоит из нескольких уровней (рис. 1). На нижнем находится родительский процесс (Host Process). Взаимодействие с родительским процессом осуществляется посредством родительского уровня (Hosting Layer - Workflow). Его интерфейсы для ключевых процессов уровня исполнения WWF должны реализовать разработчики, интегрирующие WWF в свои приложения. Ядром технологии WWF является уровень исполнения (Runtime Engine), который отвечает непосредственно за workflow-моделирование и содержит необходимые для этого службы. Он же управляет жизненным циклом (менеджмент состояний и активация) моделей. На самом верху архитектуры WWF – уровень workflow-модели (Workflow Model Layer). Он отвечает за поддержку различных типов моделей, содержит предопределенные действия (activities) и реализует соответствующий API.

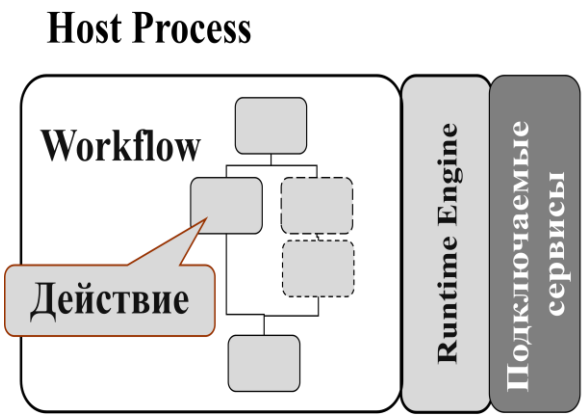

Рис. 1. Архитектура WWF

Пользовательский интерфейс среды WWF содержит встроенный визуальный редактор (рис. 2), благодаря которому можно конструировать модели, автоматически сохраняемые в XOML-файлах. При необходимости модели потоков работ могут быть реализованы исключительно посредством программного кода, поскольку фактически представляют собой обычные классы.

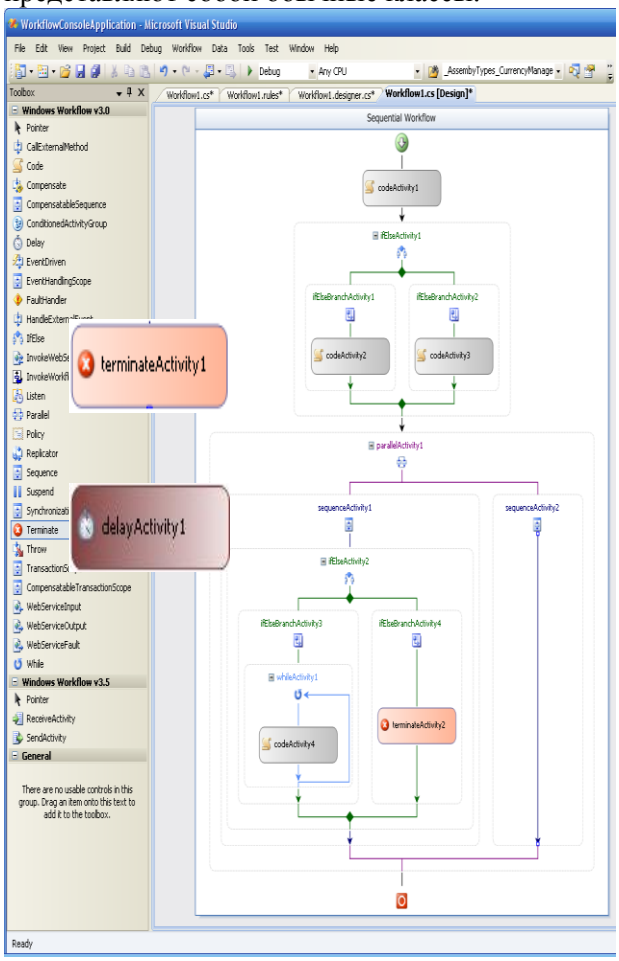

Рис. 2. Последовательная модель в WWF

Одной из главных особенностей, характеризующих технологию WWF, является гибкая способность к расширению, как уже существующих функциональных моделей, так и путём создания новых активностей. Отдельные элементы workflow-модели – действия (activities) – выполнены в виде компонентов, которые могут создаваться сторонними поставщиками с помощью Visual Studio. Помимо этого, WWF поддерживает динамическое обновление моделей во время исполнения, а также позволяет встраивать редактор моделей в собственные приложения. WWF поддерживает два типа моделей – последовательные – sequential (рис. 2) и конечные автоматы – state machine. Первые представляют собой набор последовательно исполняемых действий и в наибольшей мере соответствуют классическим блок-схемам. Вторые позволяют моделировать переходы между несколькими предопределенными состояниями, что особенно актуально при конструировании сложных программных систем, например управления бизнес-процессами [9]. Базовый набор активностей состоит из 28 активностей (на рис.2 слева) и включает в себя возможности управления исполнением, построения циклов, распараллеливания, работу с исключениями, подсоединение к источникам данных, связывание с Web-службами.

Исполняющая система Runtime Engine, предоставляет следующие службы запущенным потокам работ:

– *служба постоянства* – позволяет сохранять текущее состояние потока, длительное время находящегося в состоянии простоя с последующим возобновлением активностей;

– *служба отслеживания* – записывает события потока, которые были определены пользователем для отслеживания;

– *пользовательские службы* – создание пользовательских служб путём реализации интерфейса IDoorService.

Поддерживается возможность изменение потока работ во время работы (on-the-fly) путём создания объекта WorkflowChanges, содержащего все новые действия, которые нужно добавить к потоку работ, и вызова ApplyWorkflowChanges, определенный в классе WorkflowInstance, для фиксации этих изменений.

Конструктор Workflow Designer, используемый для проектирования потоков работ, не привязан к Visual Studio – при необходимости он может быть развёрнут в среде своего собственного приложения. Это даёт возможность поставлять систему, включающую потоки работ, и позволяющую пользователям настраивать ее под свои нужды. Можно предоставить пользователю пустой рабочий поток в качестве шаблона и обеспечить панелью инструментов, включающей специальные действия, которые отвечают предметной области. Затем, пользователи самостоятельно смогут

конструировать свои потоки работ, добавляя эти действия или разрабатывая собственные.

Таким образом, технология WWF позволяет разрабатывать как законченные пользовательские приложения, так и заготовки, внутренняя логика которых в дальнейшем может быть изменена самим пользователем без

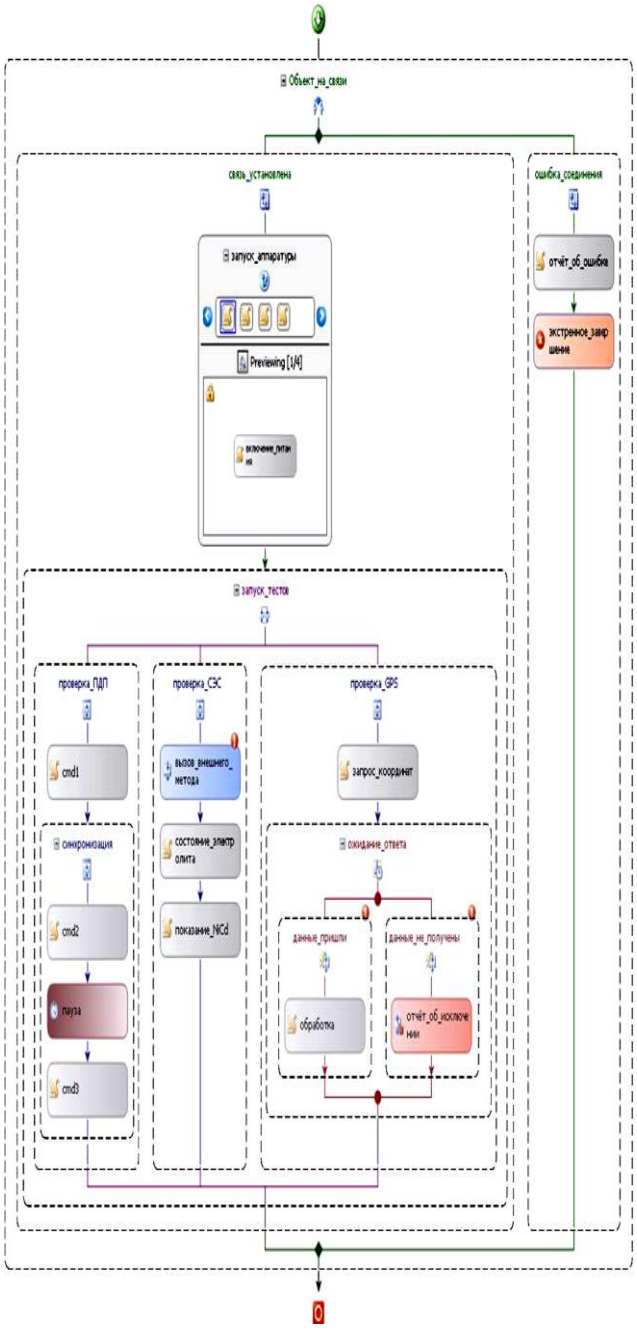

Рис. 3 Описание алгоритма стендовых испытаний КА с помощью WWF в Visual Studio 2008

перекомпиляции всего проекта через встроенный редактор потоков работ. Эффективность применения данной технологии обеспечивается многими факторами:

– возможностями создания действий, характерных предметной области и использования шаблонов действий;

– использованием графического редактора на этапе разработки и, при необходимости, встраиваемого в пользовательский интерфейс;

– визуализацией хода исполнения, возможностями сохранения состояний и отслеживания событий и т.д.

3.2 Пример реализации сценария технического процесса испытаний в среде WWF

На рис.3 представлен упрощённый фрагмент ПО, реализующий проведение стендовых испытаний КА. Сначала проводится проверка связи с текущим объектом испытаний, если связь установить не удаётся, то происходит формирование отчёта об ошибке и генерация исключения. Иначе, запускается процесс испытаний, состоящий из нескольких этапов: сначала запуск аппаратуры (с ожиданием включения всех подсистем), потом параллельное обращение к различным подсистемам спутника – подсистеме данных платформы (ПДП), системе энергоснабжения (СЭС), GPS навигатора.

Схема потоков работ, изображённая на рисунке 3, содержит следующие логико-структурные элементы:

– *объект_на_связи* – составное действие типа **IfElseActivity**, реализует выбор между ветвями *связь_установлена* и *ошибка_соединения* типа IfElseBranchActivity. Каждая ветвь также является составным действием, унаследованным от SequenceActivity – класса, выполняющего последовательность действий одно за другим;

– *отчёт_об_ошибке* – элемент типа **CodeActivity**, содержит набор инструкций, для генерации отчёта;

– *экстренное_завершение* – активность типа **TerminateActivity** осуществляет выход из приложения;

– *запуск_аппаратуры* – составное действие типа **ConditionedActivityGroup**, содержит ряд дочерних действий (возможны составные действия и установка дополнительных условий выполнения на каждое дочернее действие) и выполняет их,

– пока глобальное условие не станет истинным, т.о комбинирует поведение действий While и IfElse;

– *запуск_тестов* – составное действие типа **ParallelActivity** позволяет определить набор действий, выполняемых параллельно; ветках типа **SequenceActivity** (*проверка_ПДП, проверка_СЭС, проверка_GPS*), которые включают в себя набор других действий;

– *синхронизация* – элемент типа **SynchronizationScopeActivity**, обеспечивает непрерывность части рабочего процесса, при входе в этот блок будут непрерывно выполнены: команда *cmd2*, выдержана необходимая *пауза* (элемент типа DelayActivity) и *cmd3;*

– *вызов_внешнего_метода* – активность типа **CallExternalMethodActivity** обеспечивает вызов внешних методов или внешних служб;

– *ожидание_ответа* – активность типа **ListenActivity** обеспечивает организацию ожидания одного из набора возможных событий с указанием максимального времени ожидания;

– *отчёт_об_исключении* – активность типа **ThrowActivity** используется для генерации исключения, для перехвата исключений, выданных действиями рабочего процесса, можно добавить действие FaultHandler конструкторы рабочего процесса предоставляют контейнер для всех обработчиков ошибок.

3.3 Метод верификации моделей WWF на основе сетей Петри

Сеть Петри представляет собой ориентированный двудольный граф с двумя типами вершин, которые называются позициями и переходами. Вершины соединены направленными дугами. Вершины одного и того же типа не могут быть соединены дугой. Графически позиции изображаются кругами, а переходы – прямоугольниками [5].

Сеть Петри определяется как тройка (P,T,F), где:

P – конечное множество позиций;

T – конечное множество переходов ($P \cap T = \varnothing$);

$F \subseteq (P \times T) \cup (T \times P)$ – множество дуг (определяющих направление потока в сети).

В каждый момент времени позиция содержит нуль или более фишек, изображаемых точками. В процессе исполнения сети число фишек может меняться. Переходы являются активными компонентами сети Петри. Они меняют состояние сети в соответствии со следующими правилами срабатывания:

1) переход t называется активным, если каждая входная позиция для t позиция p содержит по крайней мере одну фишку;

2) активный переход может сработать. Если переход t срабатывает, то t забирает по одной фишке из каждой входной для t позиции и помещает по одной фишке в каждую выходную для t позицию.

Введём следующие условные обозначения:

p – входная позиция перехода t, если в сети имеется дуга, направленная от p к t;

$\bullet t$ – множество входных позиций перехода t*;*

$t\bullet$ –множество выходных позиций;

$\bullet p$ – множество переходов, для которых *p* является входной позицией;

$p\bullet$ – множество переходов, для которых *p* является выходной позицией;

M – состояние сети, называемое разметкой, определяется, как распределение фишек по позициям;

$M_1 \xrightarrow{t} M_2$ – переход t является активным в состоянии M_1 и срабатывание t переводит состояние M_1 в состояние M_1*;*

(PN, M) – сеть Петри PN с начальным состоянием M;

В среде разработки WWF построение Workflow модели осуществляется путём моделирования последовательной, условной, параллельной и итеративной маршрутизации, используя базовый набор активностей, состоящий из 28 компонент, которые можно представить в виде элементарных функциональных блоков И-разветвление, И-слияние, ИЛИ-разветвление и ИЛИ-слияние. Если для представления и анализа потоков работ, построенных по спецификации WWF, использовать сети Петри, то задачи следует моделировать переходами, а причинные зависимости позициями и дугами.

Позиция соответствует условию, которое может использоваться в качестве пред- и/или постусловия для задач. И-разветвлению соответствует переход с двумя и более выходными позициями, а И-слиянию – переход с двумя или более выходными позициями. ИЛИ-разветвлению и ИЛИ-слиянию соответствуют позиции с несколькими выходными /входными дугами. Ниже

рассмотрены способы преобразования базовых активностей WWF к нотации сетей Петри.

Определение **WF-сети** [5]: Сеть Петри PN=(P,T,N) называется WF- сетью (сетью потоков работ), тогда и только тогда, когда выполняются следующие три условия:

*1)*имеется одна позиция-источник $i \in P$, такая что $\bullet i = \varnothing$;

*2)*имеется одна позиция-сток $o \in P$, такая что $o\bullet = \varnothing$;

3)каждая вершина $x \in P \cup T$ находится на некотором пути от позиции **i** к позиции **o** (добавлено для исключения “висячих” задач).

Сеть потоков работ имеет одну входную позицию (**i**) и одну выходную позицию (**o**) , так как каждый экземпляр WF-сети создаётся в момент появления в системе управления потоками работ и удаляется после полной обработки. Таким образом, WF-сеть описывает жизненный цикл экземпляра.

Современные исследователи выделяют из 20 основных паттернов потоков работ [9] 6 групп, при помощи которых можно смоделировать любой поток:

1) базовые паттерны (присутствуют в большинстве языков потоков работ для моделирования последовательной, параллельной и условной маршрутизации);

2) расширенные переходы и синхронизации (расширяют базовые до более сложных случаев разветвления и объединения, примером может служить синхронизирующее соединение, который действует либо как XOR-слияние, либо как AND-слияние, в зависимости от контекста);

3) структурные паттерны (произвольные циклы, неявное завершение);

4) множественные экземпляры (применяются для выполнения некоторой части процесса несколько раз);

5) паттерны состояний (отложенный выбор, параллельная маршрутизация с чередованием, контрольные точки);

6) паттерны отмены (позволяют организовать отмену от одного действия до всего экземпляра).

Основу классификации, предоставляемых WWF, составляют две группы: простые и составные активности. К простым активностям относят те, которые состоят из одной позиции, например – вызов внешнего метода, активность встроенного кода, таймер задержки, активность прерывания. К составным активностям относят сложные структуры, такие как активности ветвления (*while*, *ifelse*), активности распараллеливания и синхронизации потоков работ и прочие. Все эти активности реализуются с помощью вышеперечисленных шаблонов, которые однозначно переводятся в нотацию сетей Петри.

На рис. 4 представлен пример преобразования *ListenActivity (*активность прослушивания*)* WWF к нотации сетей Петри.

Активность прослушивания – это некий аналог активностей *IfElse* и *Parallel* и применяется для работы с событиями. Она может содержать два или более контейнеров-последовательностей (*EventDriven),* каждый из которых содержит произвольный набор вложенных активностей.

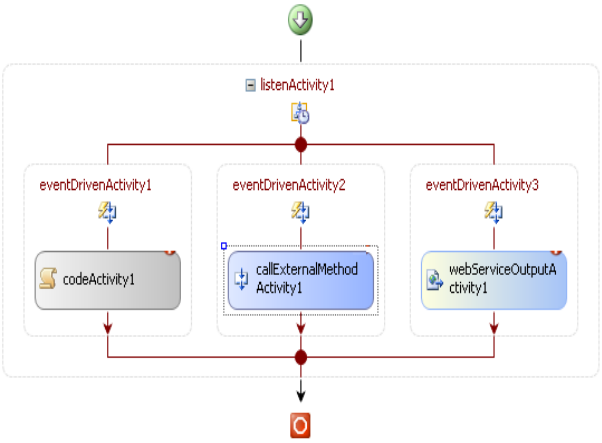

Рис. 4. Реализация ListenActivity в WWF.

Все *EventDriven* ветви в *Listen* ожидают прихода сообщения для события, на которое они подписаны. Как только приходит сообщение, соответствующее какой-то из ветвей, начинает выполняться соответствующая *EventDriven* последовательность.

Данную активность можно промоделировать при помощи паттерна “*множественный выбор*”, который представлен на рисунке 5.

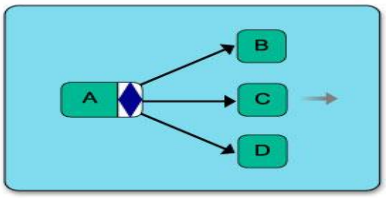

Рис. 5. Реализация паттерна “множественный выбор в нотации YAWL.

После выполнения активности *A* будет выбрана либо активность *C*, либо активность *B* либо и та, и другая, порождая при этом параллельные потоки. Это решение будет принято в зависимости от условий, которые сосредоточены в этих активностях.

Если условие не срабатывает, тогда необходимо избавиться от меток. На рисунке 6 показан механизм избавления от меток при моделировании сетями Петри.

Для избавления от ненужных меток можно на каждое условие добавить параллельную пустую ветвь, ведущую в конечную позицию сети. То есть, метка будет продвигаться до тех пор, пока не встретится конструкция синхронизации.

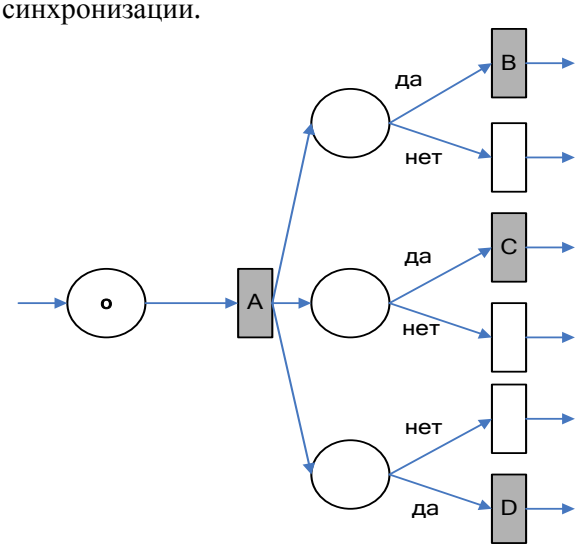

Рис. 6. Реализация паттерна “множественный выбор” и ListenActivity в нотации сетей Петри

3.4 Свойства WWF, подлежащие верификации

Для осуществления формальной верификации WWF модели необходимо учитывать следующие структурные свойства соответствующей ей сети Петри.

Живость. Сеть Петри (PN, M) называется *живой*, если для любого достижимого состояния **M’** и любого перехода **t** существует достижимое из **M’** состояние **M’’**, в котором **t** является активным.

Ограниченность. Сеть Петри (PN, M) является ограниченной, если для любой позиции существует натуральное число **n**, такое, что в любом достижимом состоянии число фишек в позиции **p** не превышает **n**.

Безопасность представляет собой частный случай ограниченности, при этом максимальное число фишек в любой позиции этой сети не превышает 1.

Бездефектность. При верификации **WF-сети** условия, перечисленные в определении WF-сети, представленном в пункте 3.3 можно проверить статически, т.е. анализируя структуру сети Петри. Но есть и ещё одно условие, которое должно быть выполнено: процедура обработки любого экземпляра, в конце концов, завершается, и в момент завершения позиция **o** содержит одну фишку, а все остальные позиции являются пустыми.

Также не должно быть “мёртвых” задач, т.е. необходимо, чтобы каждую задачу можно было выполнить, следуя подходящему маршруту в WF-сети.

Два этих требования и составляют *свойство бездефектности,* которое помимо всего прочего характеризует динамику сети.

Моделируемая WF-сетью PN=(P,T.F) процедура называется бездефектной, если:

1) для любого состояния **M**, достижимого из состояния **i**, существует последовательность срабатываний, переводящая состояние **M** в состояние **o**:

$$\forall_M (i \xrightarrow{*} M) \Rightarrow (M \xrightarrow{*} o) \qquad (1)$$

2) состояние o является единственным состоянием, которое достижимо из состояния **i** и содержит хотя бы одну фишку в позиции **o**:

$$\forall_M (i \xrightarrow{*} M \ \wedge M \geq o) \Rightarrow (M = o) \qquad (2)$$

3) в сети (PN, i) нет мёртвых переходов:

$$\forall_{t \in T} \exists_{M,M'} i \xrightarrow{*} M \xrightarrow{t} M' \qquad (3)$$

Для определения бездефектности WF-сети PN=(P,T,N) необходимо связать живость и ограниченность. Для этого определим расширенную сеть $\underline{PN}=(\underline{P},\underline{T},\underline{N})$, которая получается добавлением нового перехода t*, соединяющего позиции **o** и **i**.

Расширенная сеть Петри $\underline{PN}=(\underline{P},\underline{T},\underline{N})$ определяется следующим образом:

$\underline{P}=P$, $\underline{T}=T\cup\{t^*\}$ и $\underline{F} = F\cup\{\langle o,t^*\rangle,\langle t^*,t\rangle\}$ (4)

Такую расширенную сеть ещё называют замыканием сети PN, для которой справедлива следующая *теорема*: WF-сеть является бездефектной в том и только в том случае, когда сеть ($\underline{PN}$, i) – живая и ограниченная. Доказательство теоремы приведено в источниках [10, 11]. Следствием из теоремы является то, что для проверки бездефектности можно использовать стандартные методы анализа сетей Петри.

Выводы

Технология WWF открывает широкий спектр возможностей, для реализации которых ранее приходилось создавать сложные программные комплексы. Применение WWF при создании ПО для автоматизации стендовых испытаний малых КА позволило:

– предоставить технологу возможность самостоятельного формирования и повторного использования тестовых наборов команд без дополнительного привлечения программиста;

– вынести описательную часть хода испытаний во внешний источник информации;

– обеспечить модификацию алгоритма на этапе проведения испытаний;

– предоставить возможность сохранения состояний потоков работ с последующим возобновлением испытаний;

– повысить наглядность разработанного алгоритма тестирования;

– улучшить документируемость хода испытаний путём отслеживания событий.

При использовании WWF имеется возможность создать словарь действий и шаблонов, ориентированных на конкретную проблему, при этом трудозатраты на составление ПО значительно уменьшаются, повышается качество конечного продукта и не требуется особой квалификации программиста от разработчика, он может быть экспертом той отрасли, для которой создаётся программное обеспечение.

Результаты исследования доказывают возможность формальной верификации процессов workflow на основе математического аппарата сетей Петри. Основным свойством WF-сети Петри для верификации потоков работ является составное свойство – бездефектность. Показано, что преобразование моделей потоков работ в нотации сетей Петри является однозначным и непротиворечивым. Представленные модели и методы в дальнейшем планируется использовать как основу инструментальных средств для анализа потоков работ и повышения качества программного обеспечения гарантоспособных систем.

Литература

1. *Горбулін В. П., Шевцов А. І. Збереження* статусу ракетно-космічної держави – національне завдання України. Стратегічні пріоритети, №1(6), 2008 р. С 144-152

2. Языково-ориентированное проектирование программного обеспечения для автоматизации стендовых испытаний подсистемы данных платформы спутника МС-2-8/*Михнич Б.Б., Олейник С.В., Соколова Е.В.* // Радиоэлектроные и компьютерные системы –2008–5(32) –С.173-176.

3. Солнечные энергосистемы космических аппаратов. Физическое и математическое моделирование/ Под ред. акад. НАН Украины С.Н. Конюхова. – Харьков: Гос. аэрокосмический ун-т "Харьк. авиац. ин-т", 2000. – 516 с.

4. *Советов Б. Я., Яковлев С. А.* Моделирование систем. М.: Высш. шк., 2001. 322 с.

5. *Касьянов В. Н., Евстигнеев В. А.* Графы в программировании: обработка, визуализация и применение. СПб.:БХВ-Петербург, 2003. 262с.

6. *Baeten J. C. M., Bergstra J. A.* Real time Process Algebra. Formal Aspects of Computing, Vol. 3, 1991. P.142–188.

7. *Ben-Ari M., Manna Z., Pnueli A.*: The Temporal Logic of Branching Time. Proc. 8th Annual Symposium on Principles of Programming Languages ACM Press, Williamsburg. Springer-Verlag, 1992. P. 164-176.

8. C# 2005 и платформа .NET 3.0 для профессионалов / *Кристиан Нейгел, Билл Ивьен* – Издательство: Диалектика, 2008 – 1376 страниц.

9. Yet Another Workflow Language [Электронный ресурс]/режим доступа:*http://www.yawl-system.com/*

10. *Wil van der Aalst* Workflow Management: Models, Methods and Systems / Wil van der Aalst, Kees van Hee – Cambridge MIT Press, MA, USA 2002

11. *W.P.M. van der Aalast.* Verification of Workflows nets / Application and Theory of Petri Nets / Berlin Springer-Verlag – 1997, pages 407-426.

Сведения об авторах:

Туркин Игорь Борисович – д.т.н., професор, зав каф. инженерии программного обеспечения Национального аэрокосмического университета им. Н.Е. Жуковского «ХАИ», Харьков, Украина; научное направление – проектирование ПО систем реального времени.
e-mail: energy@d4.khai.edu.

Михнич Борис Борисович – аспирант каф. инженерии программного обеспечения Национального аэрокосмического университета им. Н.Е. Жуковского «ХАИ», Харьков, Украина; научное направление – методы формальной верификации.
e-mail: boris.mikhnich@rambler.ru.

Стаття надійшла до редакції 18.02.2010