УДК 004(4'242+053)

С.И. Чайников[1], А.С. Солодовников[2]
[1]ХНУРЭ, г. Харьков, Украина, stdep@kture.kharkov.ua;
[2]ХНМУ, г. Харьков, Украина, stdep@kture.kharkov.ua

К ВОПРОСУ ОЦЕНИВАНИЯ ПОСЛЕДОВАТЕЛЬНОСТЕЙ ВЗАИМОДЕЙСТВИЯ МОДУЛЕЙ ПРОБЛЕМНО-ОРИЕНТИРОВАННЫХ ПРОГРАММНЫХ КОМПЛЕКСОВ

Рассмотрены принципы формирования автоматной модели последовательностей взаимодействия модулей проблемно-ориентированных программных комплексов и использования соответствующего автоматного метода в целях повышения надежности программной системы на этапе формирования структуры программного комплекса. Показано использование автоматного метода в условиях динамически меняющихся требований заказчика, позволяющего повысить надежность системы при ее конфигурации.

КОНЕЧНЫЙ АВТОМАТ, BDD, МЕНЯЮЩИЕСЯ ТРЕБОВАНИЯ, ГРАФОВАЯ МОДЕЛЬ, ПРОГРАММНЫЙ КОМПЛЕКС

Введение

Надежность является важным качеством сложных программных систем (ПС), таких как, например проблемно-ориентированные программные комплексы (ПК). Сбой может привести к ошибкам, которые легко восстановить, но так же и к существенным потерям информации. Поэтому разработка таких систем должна базироваться на процессах обеспечения требуемого качества системы, верификации и валидации, и соответствующих им методах. Учитывая современные тенденции проектирования и разработки программного обеспечения (ПО) [1-3], которые ориентируются на требования заказчика, можно сказать, что процесс верификации является наиболее важным в вопросах повышения надежности систем. Особенно *актуальным* является вопрос обеспечения качества программных систем в условиях меняющихся требований.

В настоящее время для описания поведения ПС используются методы формирования программных спецификаций на естественном языке, как это принято, например, для технологии BDD (Behavior-Driven Development) или TDD (Test-Driven Development) [3], или автоматные модели и их расширения [4]. Конечные автоматы (КА), как разновидность автоматных моделей, используются для анализа возможных состояний ПС, возможных эволюций состояний, выявления недостижимых состояний и блокировок.

Модели КА обладают рядом преимуществ. Они используются для управления тем, какие наборы ресурсов следует удерживать в памяти в любой момент времени, или, какие элементы пользовательского интерфейса должны находиться на тех или иных участках экрана пользовательского приложения [5]. Базируясь на процедурном стиле программирования, позволяют описать действия подпрограмм или модулей, составляющих архитектуру ПС, и выполнять при проектировании системы функциональную декомпозицию. Данную декомпозицию выполняют различными способами, учитывая соответствующие критерии: 1) режим работы (когда за каждым вложенным автоматом может быть закреплена определенная непрерывающаяся последовательность действий); 2) по объектам управления (каждому объекту управления определяется свой вложенный автомат [6]). В последнем случае система, построенная с помощью декомпозиции по объектам управления, получается крупнее и сложнее. Однако положительный эффект декомпозиции по объектам управления заключается в использовании ее в объектно-ориентированном программировании (ООП) с явным выделением состояний. При наличии одинаковых объектов автоматизации описывают только один из них в виде класса, который затем может быть наследован [7].

Правильность функционирования КА, закладывающихся в ПС, проверяется при помощи методов верификации автоматных моделей, например, при помощи темпоральных логик LTL или CTL, а также автоматизирующих этот процесс средств, например, верификаторы «SPIN» или «Cadence SMV».

В случае формирования спецификаций требований в рамках методов, ориентированных на поведение ПС (BDD или TDD подход) процесс верификации основывается на информации о функциональности системы, получаемой из документации, описывающей статические и динамические аспекты ПС. Причем для BDD свойственно отказываться от низкоуровневых модульных тестов и переходить на спецификации поведения более высокого уровня для компонент ПО. Подход, основанный на TDD, применяется для небольших систем для проверки низкоуровневых компонент при наличии соответствия между «поведением» и компонентой и базируется на модульных тестах (или Unit-тестах). Подход, основанный на BDD, включает в себя идеи разработанных и существовавших ранее технологий в различных вариантах, например, разработки через тестирование TDD и таких подходов как «Спецификация по примеру» (Specification by Example), разработка на основе приемочных тестов (Acceptance-Test-Driven Development или Acceptance Test-Driven

Planning). Такие технологии подразумевают использование своего жизненного цикла (ЖЦ) разработки ПО, который, в основном, является эволюционным.

Учитывая преимущества автоматных моделей и гибкость BDD технологий целесообразным является решение *задачи* разработки и применения автоматного метода оценивания последовательностей взаимодействия модулей проблемно-ориентированных ПК, позволяющего в условиях меняющихся требований заказчика сформировать нефункциональные ограничения на структуру ПК и повысить надежность его функционирования.

1. Автоматный метод оценивания последовательности взаимодействия программных модулей

Входными данными автоматного метода является графовая модель структуры проблемно-ориентированного ПК, для которой определены вершины, отвечающие за формирование пользовательского диалога и за формирование соответствующих им подграфов модулей, реализующих связанные задачи системы. Графовая модель представляет собой совокупность вершин, которым сопоставляются элементарные программные модули, выполняемые последовательно или независимо друг от друга.

Ориентированные дуги между ними определяют тип связей по данным и описываются парой множеств характеристик программных модулей, которые содержат информацию о входных и выходных параметрах. Графовая модель формируется с избыточной для заданной предметной области функциональностью и обеспечивает гибкую конфигурацию системы в условиях эволюционно меняющихся требованиях. Используя данные графовой модели, формируются сценарии взаимодействия программных модулей комплекса, а определенные базовые состояния и условия переходов позволяют получить модель КА. Автоматная модель, предложенная для описания действий исполнителя вычислительных процессов (ВП) [8], позволяет выполнить процесс декомпозиции и выделить два вложенных КА, моделирующих прямой ход ВП и восстановление данных вычислений. Первый из вложенных КА представляет интерес в смысле проектирования и разработки исполнителя ВП, моделируя сценарии взаимодействия программных модулей и сравнивая результаты моделирования с нефункциональными требованиями к ПК. Обозначив вложенный КА, реализующий прямой ход ВП как «AForwardSM», зададим для него диаграмму состояний – рис. 1.

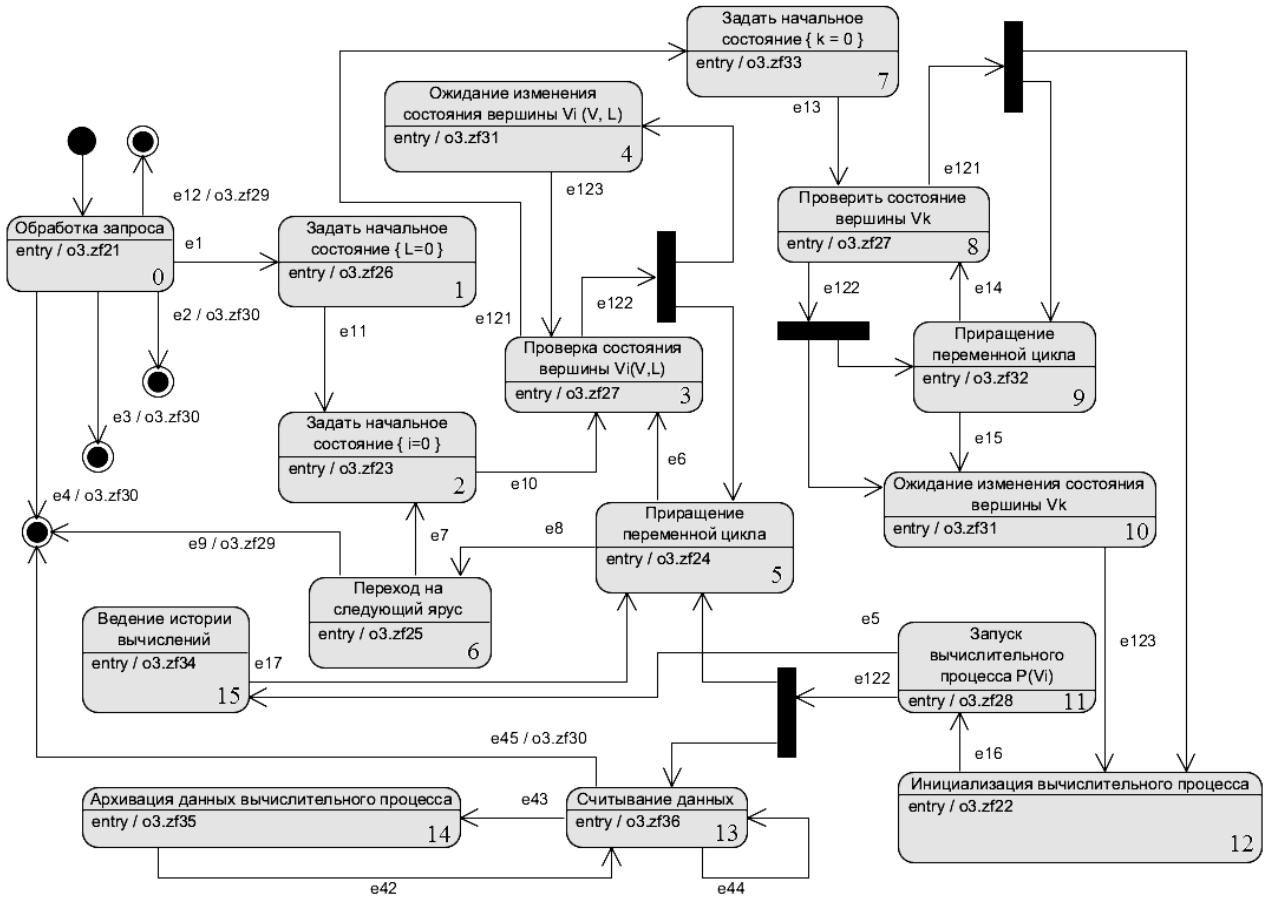

Рис. 1. Диаграмма переходов для вложенного конечного автомата AForwardSM

При этом учитывается то, что ВП, которыми они управляют, могут выполняться параллельно и независимо друг от друга (для процессов, располагающихся на одном ярусе графовой модели структуры ПК), поэтому для диаграммы КА задаются распараллеливающие и синхронизирующие переходы, обеспечивающие ввод автомата одновременно в несколько состояний. Для предложенной диаграммы выделяются две главные задачи: 1) запуск ВП; 2) архивация данных ВП. Для диаграммы принимаются следующие обозначения: V – является множеством вершин подграфа G, который определяется для текущей задачи, запускаемой пользователем; L – текущий номер яруса; P(Vi) –ВП, соответствующий вершине v_i; Vi(V, L) – означает вершину $v_i \in V$, которая находится на ярусе L; Vk – вершина v_k, из которой существует направленная дуга (v_k, v_i) в вершину v_i. В целях обеспечения моделирования последовательностей взаимодействия программных модулей выполним программную реализацию автоматной модели вложенного КА AForwardSM.

2. Практическая реализация

Закладывая возможность управления распределенными ВП, воспользуемся шаблоном распределенного выполнения программ на базе служб MCF (рис. 2), в котором функции КА верхнего уровня возложены на класс ServiceMCF, а функции вложенного КА AForwardSM возложены на класс AUserSM. Класс ServiceMCF реализует интерфейс класса IServiceMCF с обязательными функциями GetData() и GetDataUsingDataContract(), организующими обмен информацией с клиентами.

Последняя функция осуществляет прием и передачу параметров с использованием контракта данных, который используется для сериализации новых созданных сложных типов данных. Для класса ServiceMCF также задается процедура f_Call_SM() вызова управляющего КА AUserSM с его методами. Взаимодействие службы и программных модулей осуществляется посредством механизма классов-оберток. При этом классы ClientMCF_Wrp1, ClientMCF_Wrp2, ClientMCF_WrpN самостоятельно информируют службу о состоянии ВП, после чего, служба передает необходимую информацию автомату AUserSM, который обновляет состояние графовой модели.

Верификация автоматной модели осуществляется с использованием линейной временной логики (LTL), служащей для описания требований к автоматной модели, в связи с тем, что это наиболее удобный вариант, позволяющий при верификации и определении спецификаций ограничиться понятием КА.

Предложенную автоматную модель целесообразно описать программно на языке PROMELA в целях дальнейшего использования автоматизированных средств верификации свойств полученной модели. Для этого используется находящееся в свободном доступе ПО JSPIN как инструмент формальной верификации распределенных систем. Автоматная модель, реализованная на языке PROMELA, выглядит следующим образом.

```
/*AForwardSM*/
#define p1 (A1Events=12)
int A1States;
#define p2 (events[0]=23)
```

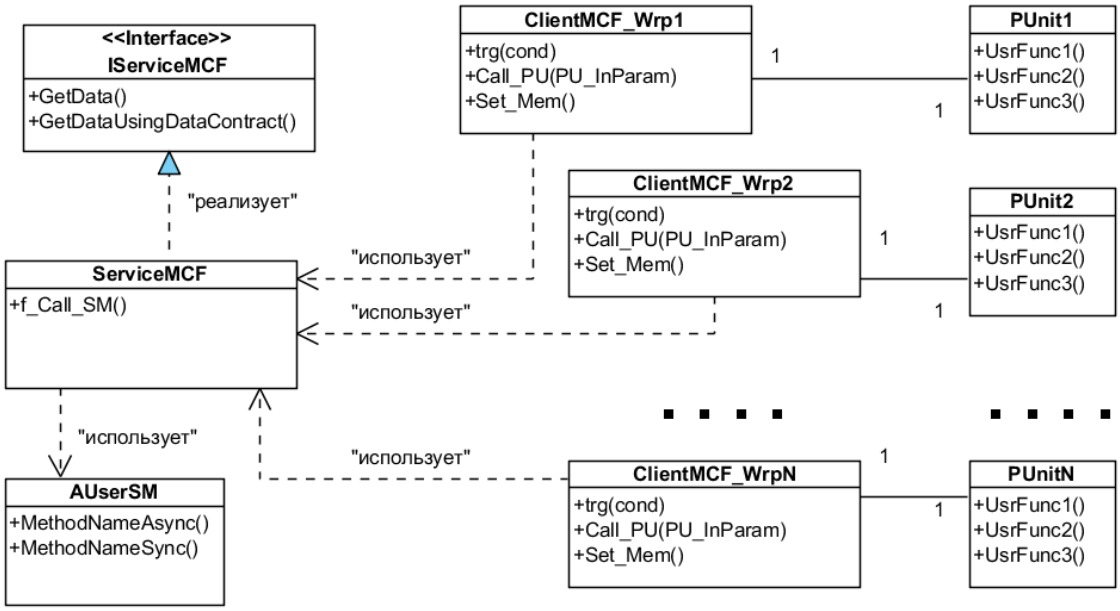

Рис. 2. Шаблон организации распределенного выполнения программы с использованием службы MCF

```
int A1Events;
ltl { !(<>((A1Events==12)) V (A1States==16)) }
proctype A1()  {
A1States=0;
A1Events=0;
      do
      ::(A1States==0) ->
                  do
                              ::A1States=1; A1Events=1;
                              ::A1States=16; A1Events=2;
::A1States=16; A1Events=3;
::A1States=16; A1Events=4;
::A1States=16; A1Events=12;
                              ::break
                  od;
      ::(A1States==1) ->
                  A1Events=11;
                  A1States=2;
      ::(A1States==2) ->
                  A1Events=10;
                  A1States=3;
      ::(A1States==3) ->
                  do
                              ::A1Events=122;A1States=4;
                  ::A1Events=122;A1States=5;
                              ::A1Events=121;A1States=7;
                              ::break;
                  od;
      и т. д.
```

При этом определяются следующие переменные: A1States – номер текущего состояния автомата, а A1Events – номер события, при котором автомат переходит в это событие. Начальные значения этих переменных равны нулю. Затем определяется процедура A1(), описывающая правила перехода автомата из одного состояния в другое. Внутри этой процедуры определяется цикл do, в рамках которого процедура проверяет текущее значение переменной A1States и, в зависимости от значения, переходит в соответствующий вложенный цикл. Во вложенном цикле также перебираются все возможные переходы. При этом, переменной A1Events задается номер события, которое послужило причиной этого перехода. Таким образом происходит описание всех состояний и переходов заданного КА (рис. 1), что позволяет затем провести автоматическую верификацию свойств автоматной модели.

Выводы

Предложенная автоматная модель позволяет сформировать ограничения к структуре ПК с учетом меняющихся требований и повысить надежность его функционирования, опираясь на данные графовой модели структуры проблемно-ориентированного ПК. При этом свойства автоматной модели формализуются на базе темпоральной логики LTL, дополняются новыми свойствами в случае появления очередных версий нефункциональных требований к ПК и верифицируются при помощи существующих автоматизированных ПС. Предложенная автоматная модель используется для проектирования исполнителя ВП и позволяет в рамках процесса моделирования последовательностей взаимодействия программных модулей сравнить результаты моделирования с нефункциональными требованиями к ПК.

Список литературы: **1**. Лаврищева, Е.М. Software Engineering компьютерных систем. Парадигмы, технологии и CASE-средства программирования / Е.М. Лаврищева.– К.: Наук. думка. – 2013. – 283 с. **2**. Лаврищева, Е.М. Сборочное программирование. Основы индустрии программных продуктов / Е.М.Лаврищева, В.Н.Грищенко. – 2-изд. доп. и перераб. – К.: Наук. думка, 2009. – 372 с. **3**. Amodeo, E. Learning Behavior-driven Development with javascript + Code / E. Amodeo – Birmingham:Packt Publishing, 2015. – 392 p. **4.** Немченко, В.П. Моделирование сетевых протоколов при построении тестовых последовательностей / В.П. Немченко, А.Н. Зиарманд, Ю.А. Чепелев // Інформаційно-керуючі системи на залізничному транспорті. – 2011. – №5. – С. 18-21. **5**. Салмре, И. Программирование мобильных устройств на платформе .NET Compact Framework / И. Салмре, пер. с англ. – М.: Изд. дом «Вильямс». – 2006. – 736 с. **6.** Туккель, Н.И., Шалыто А.А. SWITCH-технология – автоматный подход к созданию «реактивных» систем // Программирование. - 2001. - № 5. - С. 45-62. **7**. Поликарпова, Н.И. Автоматное программирование / Н.И. Поликарпова, А.А. Шалыто. – СПб.: «Питер». – 2008. – 167 с. **8**. Чайников, С.И. Организация вычислительных процессов для заданной предметной области с использованием модели конечных автоматов / С.И. Чайников, А.С. Солодовников // Системи обробки інформації. – 2016. – № 2(139). – С. 132-137.

Поступила в редколлегию 24.05.2016

УДК 004(4'242+053)

До питання оцінювання послідовностей взаємодії модулів проблемно-орієнтованих програмних комплексів / С. І. Чайніков, А.С. Солодовніков // Біоніка інтелекту: наук.-техн. журнал. – 2016. – № 1 (86). – С.145-148

В статті розглянуто процес формування автоматної моделі послідовностей взаємодії модулів проблемно-орієнтованих програмних комплексів, у межах якого створюється програмна реалізація моделі, що верифікується за допомогою інструмента JSPIN та темпоральної логіки LTL.

Іл. 2. Бібліогр.: 8 найм.

UDC 004(4'242+053)

To the question of estimation of modules interaction sequences in application-oriented bundled software / S. I. Chainikov, A. S. Solodovnikov // Bionics of Intelligence: Sci. Mag. – 2016. – № 1 (86). – P.145-148

Authors propose the process of forming of state machine model used for modeling of modules interaction sequences in application-oriented bundled software. In scope of such a process there is the program realization of this model by temporal logic LTL and application JSPIN.

Fig. 2. Ref.: 8 items.