А.А. Клебанов, О.Г. Степанов, А.А. Шалыто

ПРИМЕНЕНИЕ ШАБЛОНОВ ТРЕБОВАНИЙ К ФОРМАЛЬНОЙ СПЕЦИФИКАЦИИ И ВЕРИФИКАЦИИ АВТОМАТНЫХ ПРОГРАММ

1. ВВЕДЕНИЕ

Автоматное программирование [1] – это метод разработки программного обеспечения (ПО), основанный на расширенной модели конечного автомата. В рамках данного подхода программы представляются системой автоматизированных объектов управления, логика поведения которых задается системой взаимодействующих управляющих автоматов.

В ряде работ [2, 3] показано, что к автоматным программам хорошо применима верификация на модели (*Model Checking*) [4]. Суть такой верификации состоит в проверке соответствия модели с конечным числом состояний (*структуры Крипке*) формальной спецификации, заданной в виде набора формул темпоральной логики. При верификации преимуществом автоматного подхода перед традиционными подходами к разработке ПО является высокая степень автоматизации, так как в автоматных программах модель поведения задается априори. Разработаны методы [5 – 7], позволяющие автоматически преобразовывать как управляющие автоматы в модель, пригодную для верификации, так и построенный верификатором контрпример в автоматную модель. Однако как при верификации автоматных программ, так и при верификации программ общего вида, существует следующая проблема – необходимость записи формальных требований в виде формул темпоральных логик, работа с которыми достаточно трудоемка и требует значительной математический подготовки.

В работе [8] эта проблема частично решается использованием контрактов [9], которые являются более простым формализмом, однако значительно уступают темпоральным логикам в выразительных возможностях. Контракты к записи требований применим только в том случае, когда необходимо специфицировать свойства инвариантности, предусловия или постусловия.

В настоящей работе описывается подход к записи требований, скрывающий сложность темпоральных логик. Предлагается записывать требования на подмножестве естественного языка (*controlled natural language*),

заданного приводимой ниже формальной грамматикой. Грамматика основывается на наборе шаблонов требований [10, 11] – обобщенном описании (формальном и на естественном языке) часто встречающихся ограничений на допустимые последовательности состояний в модели системы с конечным числом состояний. Таким образом, для каждого полученного требования существует эквивалентная формальная запись, позволяющая осуществить верификацию.

Актуальность применения шаблонов требований в контексте автоматного программирования отмечается в работе [2]: «… важным является вопрос о шаблонах (структуре) темпоральных свойств, наиболее применимых и адекватных для верификации автоматных программ. Наличие таких шаблонов позволяло бы говорить о классах темпоральных свойств автоматных моделей, что, несомненно, облегчало бы построение технологической схемы проверки автоматных программ на корректность относительно спецификации». Однако в указанной работе выделяется только одно требование, являющееся частным случаем существующего шаблона, и дальше этот вопрос никак не прорабатывается.

В начале настоящей работе кратко описываются шаблоны требований, затем приводится анализ их применимости к спецификации автоматных программ, и, наконец, вводится формальная грамматика для записи требований. В заключении сделаны выводы по работе.

2. ШАБЛОНЫ ТРЕБОВАНИЙ

В работах [10, 11] предлагается система шаблонов требований, разработанная на основе спецификаций для программ общего вида. Ниже предлагается адаптация этой системы для автоматного программирования.

Шаблоны можно классифицировать в соответствии с иерархической структурой, основанной на их семантике. В настоящее время выделено восемь основных шаблонов («Отсутствие», «Существование», «Всеобщность», «Ограниченное существование», «Предшествование», «Ответ», «Цепное предшествование», «Цепной ответ»), которые делятся на две группы – «Наличие» и «Порядок». В группу «Наличие» входят шаблоны, описывающие наличие или отсутствие состояний, в которых выполняется заданное требование. В группу «Порядок» – описывающие порядок состояний.

Описание шаблона состоит из его имени (или списка имен), цели, записи на различных формализмах (LTL, CTL и т.п.), примера использования и связи с другими шаблонами.

Каждое требование имеет *ограничение (scope)* – ту часть пути исполнения, на котором это требование должно выполняться. Всего выделены пять видов ограничений:

- Глобально – на всем пути исполнения.
- До – на пути до заданного состояния.
- После – на пути после заданного состояния.
- Между – на пути между двумя заданными состояниями.
- После-до – аналогично ограничению «Между», однако наличие правой границы интервала не является обязательным.

Отметим, что стандартно для формализмов, ориентированных на состояние, интервал, на котором должно выполняться требование, замкнут на левом конце и открыт на правом.

В табл. 1 приведен пример шаблона «Всеобщность».

Таблица 1

Шаблон «Всеобщность»

Цель		Используется для описания части пути исполнения системы, в которой содержатся *только те* состояния, в которых выполняется необходимое требование. Известен также как «Впредь» и «Всегда».	
Запись	LTL	Ограничение	Запись
		Глобально	□(P)
		До R	◊R → (P U R)
		После Q	□(Q → □(P))
		Между Q и R	□((Q & !R & ◊R) → (P U R))
		После Q до R	□(Q & !R → (P W R))
	CTL	Ограничение	Запись
		Глобально	AG(P)
		До R	A[(P | AG(!R)) W R]
		После Q	AG(Q → AG(P))
		Между Q и R	AG(Q & !R → A[(P | AG(!R)) W R])
		После Q до R	AG(Q & !R → A[P W R])

Пример использования	Этот шаблон может быть применен для описания общих свойств модели в целом или отдельной группы состояний. Например, в том случае, когда необходимо выразить свойство вида: «Если автомат находится в состоянии S, то верно P».
Связь с другими шаблонами	Этот шаблон тесно связан с шаблонами «Отсутствие» и «Существование». Наличие состояния, в котором выполняется требование, может рассматриваться как отрицание его отсутствия.

3. ПРИМЕНИМОСТЬ ШАБЛОНОВ ТРЕБОВАНИЙ К СПЕЦИФИКАЦИИ АВТОМАТНЫХ ПРОГРАММ

Рассмотрим вопрос применимости шаблонов требований к формальной спецификации автоматных программ. Для этого проанализируем требования к различным программам, доступным на сайте [12], и проверим, как они выражаются при помощи шаблонов. Пример организации промежуточных результатов анализа приводится в табл. 2. В столбцах «Требование» и «Исходная формальная запись» приводятся оригинальные требования из источника (столбец «Источник»), записанные на естественном языке и одном из формализмов соответственно. В столбце «Шаблон, Ограничение» приводится запись шаблона, подстановка необходимых утверждений в который, даст эквивалентную исходной формальную запись. В случае необходимости приводится формальное доказательство эквивалентности.

Таблица 2

Анализ применимости шаблонов требований к спецификации автоматных программ

Требование	Исходная формальная запись	Шаблон, Ограничение	Источник
Если произошла поломка нагревателя или одного из клапанов, то кофеварка (основной автомат A_0) обязательно перейдет в состояние 5.	$AG((y_{31} = 4 \mid y_{32} = 4 \mid y_2 = 4) \& y_0 = 2 \rightarrow A(y_0 = 2\ U\ y_0 = 5))$	Ответ (ограниченный), Глобально $AG(P \rightarrow A(S))$, $P: (y_{31} = 4 \mid y_{32} = 4 \mid y_2 = 4) \& y_0 = 2$, $S: y_0 = 2\ U\ y_0 = 5$	2

Всего было рассмотрено 77 требований для 13 программ из 15 источников. Установлено, что 87% требований покрывается пятью шаблонами. Процентное соотношение между использованными шаблонами приведено на рис. 1, а между использованными ограничениями – на рис. 2.

Рис. 1. Процентное соотношение между использованными шаблонами

Рис. 2. Процентное соотношение между использованными ограничениями

4. МЕТОД ЗАПИСИ ВЕРИФИЦИРУЕМЫХ ТРЕБОВАНИЙ НА ЕСТЕСТВЕННОМ ЯЗЫКЕ

Существуют различные подходы к выделению формальных (верифицируемых) требований из спецификаций, записанных на естественном языке. Отметим два основных направления [13] – синтаксический разбор текстов и использование формальных грамматик, ограничивающих естественный язык.

В настоящей работе предлагается использовать формальную грамматику, фрагмент которой приводится в табл. 3. Как отмечалось выше, грамматика основывается на основных шаблонах требований и их вариантах. Это позволяет иметь запись требования, как на естественном языке, так и на любом из формализмов, запись на которых разработана для шаблонов. `Моноширинным шрифтом` выделены указатели места заполнения шаблона реальными требованиями.

Таблица 3

Фрагмент грамматики верифицируемых требований на естественном языке

<требование>	::= <ограничение> <шаблон>
<ограничение>	::= «Для любого состояния верно, что» | «До состояния, в котором Q, верно что» | «После состояния, в котором Q, верно что» | «Между состоянием, в котором Q, до состояния, в котором R, верно что» | «После состояния, в котором Q, до состояния, в котором R, верно что»
<шаблон>	::= <отсутствие> | <всеобщность> | <существование> | <ограниченное существование> | <предшествование> | <ответ> | <ограниченный ответ> | <цепное предшествование> | <цепной ответ> | …
<отсутствие>	::= «никогда не выполняется P»
…	…

В качестве примера рассмотрим требование: «Система управления кофеваркой никогда не попадет в такое состояние, в котором она не реагирует ни на события системного таймера, ни на нажатие кнопок «ОК» и «С». В автоматной модели кофеварки требованию «Никак не реагирует ни на события системного таймера, ни на нажатие кнопок «ОК» и «С» соответствует предикат `act = end`. Наречие «никогда» подсказывает, что должен быть использован шаблон «Отсутствие» с ограничением «Глобально». Выполним порождение:

<требование> → <ограничение> <шаблон> → Для любого состояния верно, что <шаблон> → Для любого состояния верно, что <отсутствие> → Для любого состояния верно, что никогда не выполняется P

Подставив вместо P реальное требование, получим искомое формальное требования на естественном языке: «Для любого состояния верно, что никогда не выполняется `act = end`». Этому требованию на языках CTL и LTL соответствуют выражения `AG(! act = end)` и `□(! act = end)`. Эти выражения в дальнейшем используются при верификации на модели.

5. ЗАКЛЮЧЕНИЕ

Запись требований в виде формул темпоральной логики является неотъемлемой частью верификации на модели. В настоящей работе в рамках автоматного программирования предложен подход, который, во-первых, зна-

чительно упрощает этот процесс, а во-вторых, минимизирует число потенциальных ошибок в самой спецификации.

СПИСОК ЛИТЕРАТУРЫ

1. **Поликарпова Н.И., Шалыто А.А.** Автоматное программирование. – СПб.: Изд-во Питер, 2009. – 176 с. http://is.ifmo.ru/books/_book.pdf
2. **Васильева К.А., Кузьмин Е.В.** Верификация автоматных программ с использованием LTL // Моделирование и анализ информационных систем. 2007. Т. 14, № 1, с. 3–14.
3. **Кузьмин Е.В., Соколов В.А.** Моделирование, спецификация и верификация «автоматных» программ // Программирование. 2008. № 1, с. 38–60.
4. **Кларк Э., Грамберг О., Пелед Д.** Верификация моделей программ. Model Checking. – М.: Изд-во МЦНМО, 2002. – 416 с.
5. **Гуров В.С., Яминов Б.Р.** Технология верификации автоматных моделей программ без их трансляции во входной язык верификатора / Тезисы научно-технической конференции «Научное программное обеспечение в образовании и научных исследованиях». СПбГУ ПУ. 2008, с. 36–40. – http://is.ifmo.ru/download/2008-02-24_jaminov_verifikazija.pdf
6. **Лукин М.А., Шалыто А.А.** Автоматизация верификации визуальных автоматных программ / Материалы XV Международной научно-методической конференции «Высокие интеллектуальные технологии и инновации в образовании и науке». СПбГПУ. 2008, с. 296, 297. http://is.ifmo.ru/download/2008-02-25_politech_tezis.pdf
7. **Kurbatsky E.** Verification of Automata-Based Programs / Proceedings of the Second Spring Young Researchers Colloquium on Software Engineering. 2008. V. 2, pp. 15–17. http://is.ifmo.ru/verification/_kurbatsky_syrcse.pdf
8. **Степанов О.Г.** Методы реализации автоматных объектно-ориентированных программ. Диссертация на соискание ученой степени кандидата технических наук. СПбГУ ИТМО, 2009. http://is.ifmo.ru/disser/stepanov_disser.pdf
9. **Мейер Б.** Объектно-ориентированное конструирование программных систем. – М.: Изд-во Русская Редакция, 2005. – 1204 с.
10. **Dwyer M.B., Avrunin G.S., Corbett J.C.** Property Specification Patterns for Finite-state Verification / Proceedings of the 2nd Workshop on Formal Methods in Software Practice. 1998.
11. **Dwyer M.B., Avrunin G.S., Corbett J.C.** Patterns in Property Specifications for Finite-state Verification / Proceedings of the 21st International Conference on Software Engineering. 1999.
12. Веб-сайт кафедры «Технологии программирования». http://is.ifmo.ru/
13. **Konrad S., Cheng B.H.C.** Facilitating the Construction of Specification Pattern-based Properties / Proceedings of the IEEE International Requirements Engineering Conference. 2005.